Modelling, instrumentation and control of a 6-DOF positioning stage

E.E. Visser

DCT 2004.112

Master's thesis

Coach(es): Prof. Dr. Ir. M. Steinbuch

Supervisor: H. Gutierrez, B.Sc. M.Eng. Ph.D.

Technische Universiteit Eindhoven
Department Mechanical Engineering
Dynamics and Control Technology Group

Eindhoven, November, 2004

Preface

This thesis describes the research I have done at the Department of Mechanical & Aerospace Engineering of the Florida Institute of Technology, Melbourne (FL). I performed this project from September 2003 until September 2004, under direct supervision of Hector Gutierrez (FIT) and with feedback from Maarten Steinbuch. It was a challenging year, and although results are different from my expectations one year ago, it has been a very valuable experience.

I would like to thank some of the people that contributed to my graduation project. First of all, I would like to thank Hector Gutierrez for his supervision during the past year. Hector motivated me in his own, unique style. He was always late or in a hurry, but very enthusiastic and positive. The fact that he overlooked many problems sometimes drove me crazy, but also taught me some valuable lessons. I really appreciate the opportunity Hector gave me to gather this experience abroad.

Furthermore, I would like to thank Rahul Agarwal for his work on the laser system, electronics and particularly the software. I want to thank Lionel Fevre for his help manufacturing the 1-DOF setup, and Dennis Miller for the time he voluntarily spent to help me getting the laser system and DSP up and running.

Many other people have supported me during my education, and were particulary important to me during the 20 months I spent abroad. I want to thank Olaf, Miriam and my high-school friends for their loyalty, and my roommates for their interest, support and especially for the necessary distraction from my work.

Finally, I would like to thank my parents for their support, financially but foremost mentally. They made it possible for me to fully enjoy my student days, and stimulated me to seize all the opportunities I got.

Abstract

The ultimate goal of this multiple-year project is development of a positioning stage that can achieve both high resolution and large travel. The design of this machine is described in [10]. Linear motors provide long-range motion in X- and Y-direction, while magnetic actuators provide small strokes in 6 DOF's with high accuracy. The system can be customized to fit different applications such as assembly of optical components and MEMS (Micro Electro Mechanical Systems) manufacturing.

Main goal for the project described in this report is design of a control system for the 6-DOF positioning stage described in [10]. Design and implementation of MIMO-control for this machine is a complex and time consuming process. Since development of a prototype is in an early stage, this process also involves assembly, instrumentation and (re)design of components. In the limited time available, this project focussed on several essential steps.

Hardware and software for the control of the linear motors have been set up. This includes a laser system for position feedback, which was redesigned, debugged and aligned. Once the linear motors were operational, system identification was performed and a linear controller was designed and implemented. The closed loop bandwidth is over 50 [Hz], while the steady state error is limited to 50 nanometer.

A 1-DOF setup was made in which an arm with 1 rotational DOF is actuated by one magnetic actuator. A theoretical force model for the magnetic actuator was derived, and experiments on the 1-DOF setup show that this model closely matches the empirical results.

Linear and nonlinear control strategies for the 1-DOF setup were simulated and implemented. Several months were lost by delayed manufacturing of new amplifiers, and by many flaws of the DSP that was used for implementation of the controller.

A linear controller was implemented with poor results. Best results were obtained using a sliding mode controller in combination with feedback linearization of the system. The closed loop system is stable for a large operating range, and able to track a low frequency reference signal. Steady state positioning error is well below 1 micron. Due to limited availability of amplifiers, a thorough analysis of control of the 1-DOF setup could not be finished. This is an essential part of the design and requires attention in further research.

In the meanwhile, a model of the 6-DOF system was derived. Based on temporary results obtained on the 1-DOF setup, a MIMO-controller was designed and simulated. Preliminary simulation results were obtained.

Considering the results so far, it seems very hard to meet the design specifications.

Samenvatting

Het uiteindelijke doel van dit meerjarige project is ontwikkeling van een positioneertafel met hoge nauwkeurigheid en groot werkgebied. Het ontwerp van deze machine wordt beschreven in [10]. Lineaire motoren zorgen voor een grote slag in X- en Y-richting, terwijl magnetische actuators zorgen voor kleine slag in 6 vrijheidsgraden. De machine is geschikt voor verschillende applicaties, zoals de assemblage van optische componenten en de productie van MEMS (Micro Electro Mechanical Systems).

Doel van het hier beschreven project is het ontwerp van een regelsysteem voor de 6-vrijheidsgraden positioneertafel beschreven in [10]. Ontwerp en implementatie van een regelsysteem voor deze machine is een complex en tijdrovend proces. Aangezien de ontwikkeling van een prototype nog in een vroeg stadium verkeerd, omvat dit proces niet alleen het modelleren en regelen van het systeem, maar ook instrumentatie, assemblage en (her)ontwerp van componenten. Vanwege de beperkte beschikbare tijd is dit project gericht op een aantal essentiële stappen die uiteindelijk moeten leiden tot het ontwerp en de implementatie van een MIMO-regelsysteem voor het 6-vrijheidsgraden positioneer-systeem.

Hardware en software voor de regeling van de lineaire motoren zijn opgezet. Dit proces omvatte onder andere het (her)ontwerpen en opzetten van een laser systeem voor positie-feedback. Zodra alle apparatuur operationeel was, kon systeem identificatie en regeling van de lineaire motoren worden uitgevoerd. Een lineaire regelaar werd ontworpen en geïmplementeerd. De closed-loop bandbreedte is ruim 50 [Hz], en de statische positie-fout is maximaal 50 [nm].

Een opstelling waarin een arm met 1 vrijheidsgraad (rotatie) wordt gedragen door een magnetische actuator werd gebruikt voor experimenten. Een theoretisch kracht-model voor de magnetische actuator werd afgeleid, en vervolgens met experimenten in deze opstelling geverifieerd. De experimentele resultaten komen goed overeen met het theoretische model.

Lineaire en niet-lineaire regel-strategieën voor deze testopstelling zijn gesimuleerd en geïmplementeerd. Enkele maanden vertraging ontstonden bij de fabricage van nieuwe versterkers, en vanwege de vele gebreken van de DSP waarop de regelaar geïmplementeerd werd. Een lineaire regeling werd geïmplementeerd, met slecht resultaat. Het beste resultaat tot nu toe werd behaald met een sliding mode-regelaar in combinatie met een feedback-linearisatie van het systeem. Het closed loop systeem is stabiel over een groot deel van de slag, en kan een laagfrequent referentie-signaal volgen. De statische positie-fout ligt ruim beneden 1 micrometer. Vanwege de beperkte beschikbaarheid van de versterkers kon een grondige analyse van de regeling van deze testopstelling niet worden voltooid. Dit is een essentieel onderdeel van het project en vraagt om meer aandacht in de toekomst.

Ondertussen werd een model voor het 6-vrijheidsgraden systeem opgesteld. Op basis van voorlopige resultaten van de regeling van de testopstelling werd een MIMO-regelaar ontworpen en gesimuleerd.

Gezien de resultaten tot dusver lijkt het erg moeilijk om aan de ontwerp-specificaties te voldoen.

Contents

Chapter 1

Introduction

As nowadays technology tends towards the development of smaller components, the high-precision manufacturing industry is continuously pursuing higher accuracy levels. Wafer positioning stages in photolithography are certainly among the most accurate and demanding high-precision systems. Strong research efforts are dedicated to reduce positioning errors to nanometer level.

According to [10], high precision positioning stages can be classified as high precision gantry systems or as photolithography wafer stages. Using two examples, [10] describes how gantry systems achieve long travels (over 1 [m] in X- and Y-direction) with an average position accuracy of 3.0 micron. This type of machine is typically used for flat panel inspection or electronic assembly. Ultra high position lithography systems achieve position accuracy in the order of 30 [nm], but with smaller range of motion in X and Y (0.35 [m]) and very small range in vertical direction.

The ultimate goal of this multiple-year project is development of a positioning stage that can achieve both high resolution and large travel. In [10] the design of this machine is described. It provides long ranges of motion with high accuracy, and can be customized to fit different applications such as assembly of optical components and MEMS (Micro Electro Mechanical Systems) manufacturing. Production costs of the proposed machine will be significantly lower than existing positioning systems, since standard machining tolerances apply for the manufacturing of the machine's components.

Main goal for the project described in this report is design of a control system for the 6-DOF positioning stage described in [10]. The design and implementation of a control system for this type of machine is a complex and time consuming process. Since development of the prototype is in an early stage, this process does not only include modelling and control but also assembly, instrumentation and (re)design of components.

In the limited time available, the project will focus on several essential steps that must form a basis for the control system that eventually will be implemented. As described in Chapter 2, the system contains linear motors for long-range motion in X and Y-direction, and magnetic actuators for small adjustments in all 6 DOF's. Chapter 3 describes system identification and control of the linear motors. In Chapter 4, a theoretical force model for the magnetic actuators is derived. This model is verified with experiments on a 1-DOF setup. In Chapter 5, linear and nonlinear control strategies for this 1-DOF system are described. Due to problems with the hardware, a good controller of the 1-DOF setup has not been obtained yet. Finally, Chapter 6 describes a 6-DOF model of the system. Based on temporary results of the 1-DOF control, a MIMO-controller was designed and simulated.

Several issues will not be addressed in this report, that are certainly non trivial. They will require attention in future research. One of those issues is the feasibility of several design principles. For example, a metrology mapping for measurements in Z, θ_x and θ_y is proposed in [10]. This is a novel approach that is still topic of research and discussion. If the principle is feasible, the implementation will require special attention (calibration, implementation of large look-up table).

Another issue are the specifications with respect to speed and accuracy in [10]. These specifications will *not* be used as target, since it is not realistic to think design and implementation of a control system meeting all design specifications can be realized within a year. Instead, control systems will be designed that achieve maximum performance (in terms of accuracy and bandwidth) on the 1-DOF setups. Only when MIMO-control of the 6-DOF system is implemented, it will become clear if the specifications come close to those specified in [10]. From that point, iterations will have to lead to improved performance in order to eventually meet the design specifications.

Chapter 2

Previous research

This chapter contains a summary of 'Design and construction of a 6 DOF positioning system with nanometer accuracy', master's thesis of L. Fevre [10]. At the end of Fevre's project, the machine described in this thesis was partially manufactured and assembled.

2.1 Technical Background

[10] shortly describes the applications of high precision positioning devices, and gives an indication of the performance of state-of-the-art devices (in terms of accuracy, stroke, etc.). The performance[1] of this particular design is summarized in Table 2.1. (Note: 1 arc-sec = 4.8345 μrad).

2.2 Machine Design

2.2.1 General assembly

An overview of the machine can be seen in Figure 2.1. The work piece is carried by a support surface (tray). This tray has a long range of motion in the horizontal plane XY (250 x 250 mm), and can be accurately positioned in 6 degrees of freedom. The four arms carrying the tray are actuated by a combination of linear motors on air bearings (for the long range motion in X and Y direction) and novel non-contact electromagnetic actuators (for small adjustments in all DOF's).

A passive magnetic suspension carries the weight of the tray. DOF's X, Y and θ_z are measured using a laser interferometer system. A set of three capacitive sensors is used to measure Z, θ_x and θ_y. Similar capacitive sensors are used to measure the relative motion of the arms with respect to the magnetic actuators.

The maximum achievable accuracy of the machine is limited by the resolution of the measurement devices. This resolution can be improved by using a mapping of the sensor target surfaces, that will be used to compensate measurements for imperfections on these surfaces.

According to [10] the machine can be rescaled to achieve larger travels, without affecting the performance. Only the manufacturing and mapping of sensor target surfaces will be more time-consuming and expensive. Most machine components can be manufactured by a standard machine shop, since machining tolerances for this system are less strict than for existing high-precision positioning systems. Only manufacturing of the granite base, interferometer mirrors and reference surface requires extreme accuracy and has to be performed by professional companies.

[1] There are 3 possible operating modes. Depending on sensor sensitivity settings, a trade-off is made between stroke and accuracy. The table shown is based on sensor settings as used during this entire project

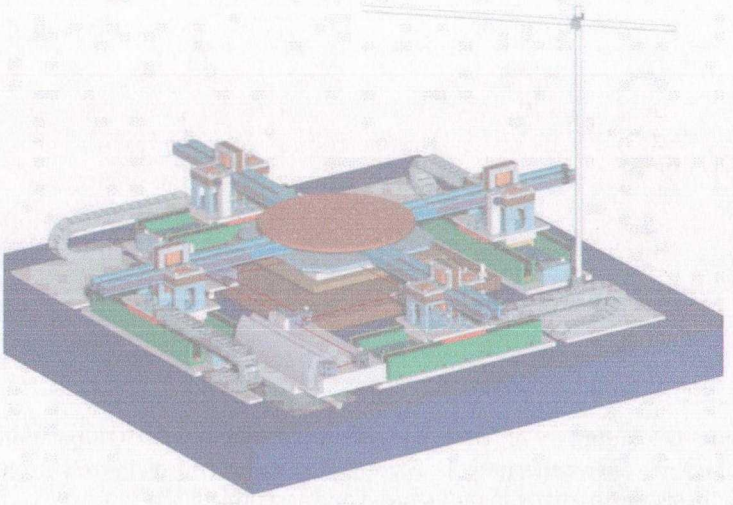


Figure 2.1: Overview of the complete positioning table [10]

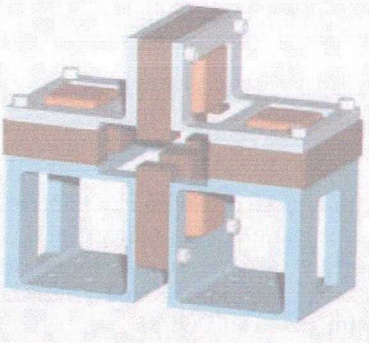


Figure 2.2: Magnet box [10]

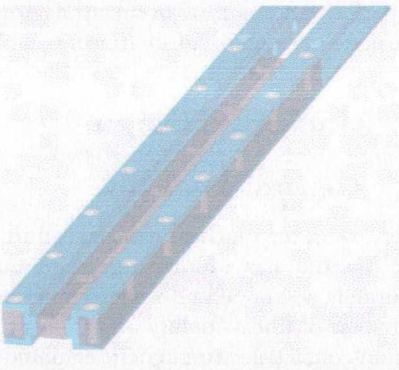


Figure 2.3: Crossmember with integrated laminations [10]

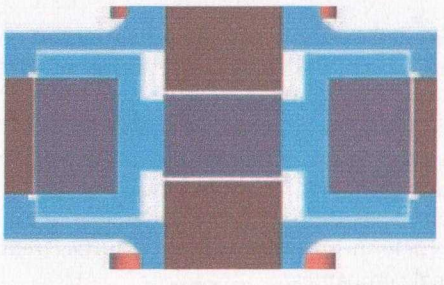


Figure 2.4: Nominal position of crossmember in magnet box [10]

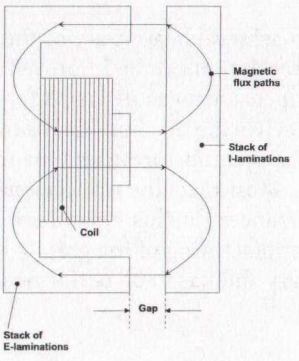


Figure 2.5: Magnetic flux paths

Parameter	Axis	Specification
Travel	X	203 mm
	Y	203 mm
	Z	1 mm
Pitch	θ_x	$\pm$ 20 arc-sec
	θ_y	$\pm$ 20 arc-sec
Yaw	θ_z	$\pm$ 20 arc-sec
Orthogonality	X to Y	$\pm$ 0.5 arc-sec
	Z to XY-plane	$\pm$ 0.02 arc-sec
Resolution	X and Y	1.25 nm
	Z	60.01 nm
	θ_x and θ_y	0.02 arc-sec
	θ_z	0.009 arc-sec
Repeatability	X and Y	1.25 nm
	Z	4.01 nm
	θ_x and θ_y	0.051 arc-sec
	θ_z	0.084 arc-sec
Accuracy	X	3.3 nm
	Y	3.3 nm
	Z	64.0 nm
Velocity	X and Y	500 mm/sec
Acceleration	X and Y	8.92 m/sec^2
	Z	12 m/sec^2
Total mass	N/A	445 kg

Table 2.1: Capabilities and performance [10]

2.2.2 Moving units

Each moving unit (see Figure 3.1) consists of a pair of linear servo motors carrying a platform with a 'magnet box'. The platform floats on an air bearing, while the linear motors move in a magnet track without making mechanical contact. Theoretically, the position resolution is only limited by the resolution of the measurement device.

2.2.3 Magnet boxes

Each electromagnetic actuator or 'magnet box' (see Figure 2.2) consists of two pairs of opposed copper coils, that are each wound around the center leg of a stack of E-shaped iron laminations. I-shaped laminations are embedded in each supporting arm of the tray (see Figure 2.3).

When current is passed through the coil, this induces magnetic flux paths that form two loops through the outer legs of the E-shaped laminations and the I-shaped laminations (see Figure 2.5). Each coil can only exert attractive force to the I-shaped laminations, so two opposite coils are needed to control one DOF. Consequently, the vertical coil pairs control the DOF's Z, θ_x and θ_y of the tray, while the horizontal pairs control X,Y and θ_z.

Based on the magnetic laws governing the system, the relations for induction and force were established.

Magnetic flux:

$$\phi = BA = \frac{NI}{\frac{l}{\mu_r \mu_0 A} + \frac{2g_n}{\mu_0 A}} \tag{2.1}$$

Induction:

$$L = \frac{N\phi}{I} = \frac{N^2}{\frac{l}{\mu_r \mu_0 A} + \frac{2g_n}{\mu_0 A}} \tag{2.2}$$

Magnetic force:

$$F = (\frac{NI\mu_r}{l + 2g_n\mu_r})^2 \mu_0 A \tag{2.3}$$

With:

μ_r	=	Relative permeability of metallic core	[-]
μ_0	=	Permeability of air	[H/m]
N	=	Number of coil windings	[-]
I	=	Coil current	[A]
A	=	Cross section of the coil	[m^2]
g_n	=	Nominal gap size	[m]
l	=	Magnetic path length	[m]

The magnetic actuators are designed to provide forces high enough to keep the tray and arms in position while the linear motors provide maximum force. This means the arms will not make contact with the magnetic actuators during maximum acceleration of the moving units. The required force was computed for a worst case situation, when the tray is as far off center as possible and the air gap is at its maximum. Additional mass (for example of an extra device to hold the work piece) was taken into account, too. Based on this computation, the magnetic actuators are designed to provide a force of at least 200 N.

The maximum coil current is limited to 3.0 A, in order to avoid high heat dissipation in the coils. Furthermore, the inductance of the coils is kept to a minimum.

The electromagnetic actuators in vertical direction are equal to those in horizontal direction, even while lower forces are required in vertical direction (a passive magnetic suspension carries the weight of the stage). The nominal air gaps in vertical direction (see Figure 2.4) are 500 μm, making the vertical range of motion 1 mm in total. Additional room is added to allow for the rotations θ_x and θ_y. The nominal air gaps in horizontal direction are 250 μm each, and allow for small rotations θ_z.

2.2.4 Tray and arms

The tray is made as stiff and light as possible, to keep its natural frequencies high and to allow a competitive acceleration of the system. The main stage consists of two plates sandwiching four center beams, to which the arms are attached (see Figure 2.6). To save material and weight, the I-shaped laminations are integrated in the arm structure. Three stacks of these laminations are assembled using two aluminum C-shaped channels (see Figure 2.3 and Figure 2.4). Apart from obvious advantages like low density, low price and easy machining, the aluminum also has a relative magnetic permeability of 1, so it is not affected by the magnetic field in the magnetic boxes.

A kinematic mounting system will be used to attach a L-shaped mirror under the tray (see Figure 2.7). This mirror is used with the interferometer system to measure the DOF's X,Y, and θ_z. More about this in Subsection 2.2.6.

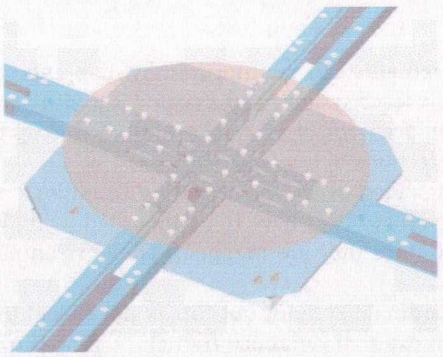


Figure 2.6: Tray assembly (top view) [10]

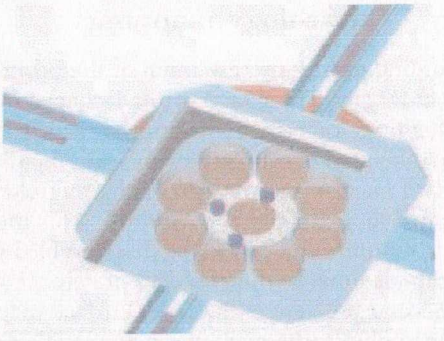


Figure 2.7: Tray assembly (bottom view) [10]

2.2.5 Passive magnetic suspension

A passive magnetic suspension provides a lifting force to the tray to compensate gravity. Consequently no bias current through the upper coils of the magnetic actuators is needed to carry the tray. A bias current leads to heat generation.

An array of equally oriented magnets generates a uniform magnetic field at the base of the machine, covering the entire operating range of the tray. An array of magnets in opposed orientation, mounted to the bottom of the tray, provides a repelling force. Main requirements to the repelling force are that it is constant over the vertical range of motion (1 mm) and over the X,Y-operating range. Peaks in the magnetic field should be avoided, because this will induce vibration in vertical direction when the tray travels in X- and/or Y-direction.

The design of this magnetic suspension has not been finished. A scale model has been made, and currently efforts are made to fit both a FEA model and analytical model to measurement data [9]. Once a good model is available, design can be finished and the suspension will be manufactured.

2.2.6 Position measurement

Two measurement frames are used to determine the position and orientation of the tray. A laser interferometer system will be used to measure the horizontal translations X and Y, and the rotation around the vertical axis, θ_z. For this purpose, a high precision L-shaped mirror is attached to the bottom of the tray (see Figure 2.7).

For the measurement of Z, θ_x and θ_y, a set of three capacitive sensors is used. These sensors are mounted to the bottom of the tray, and accurately measure distance based on the capacitance between two plates. One plate is the sensor surface, the other plate is the target which has to be a metallic surface. A target made of Zerodur (a high stiffness type of glass) with a metallic surface is recommended [10]. This target has to be manufactured with very precise flatness.

The resolution of this type of sensor depends on its capacitance. A low capacitance provides a very accurate measurement but limits the operating range of the sensor, so a trade-off has to be made. The sensors used in this project can be used in two sensitivity settings: 50 [μm/V] over a 1.0 [mm] range, or 5 [μm/V] over a 0.1 [mm] range. The resolution is 60 [nm] or 4 [nm], respectively.

Additionally, 2 capacitive sensors are used in each magnet box to measure the gaps in vertical and horizontal direction. These gaps are used in the computations for feedback control.

2.2.7 Metrology mapping

According to [10] the resolution of the position measurements can be improved by mapping of the sensor targets. These targets (the L-shaped mirror for the laser system, and a coated Zerodur surface for the three capacitive sensors) can be mapped with an accuracy of up to 0.1 [nm]. A lookup table should be used to compensate measurements for imperfections of the targets.

Two major problems arise using metrology mapping. First of all, the operating conditions must be equal to the mapping conditions. This means that both surfaces have to be mounted using kinematic mounts, and cannot be disassembled or changed once they are mapped. The environmental conditions must be controlled accurately.

Second problem is that each surface map has a coordinate system, but this is not linked to the mapped object (the mapping is done with respect to an external reference frame). A procedure for alignment of the maps' origin with the origin of the metrology frame is suggested in [10].

The proposed approach has been subject to discussion and criticism. Using feedback from [11], modifications to the design will be made before it is implemented. In the meanwhile, a temporary measurement system will be used for the design of a control system. This limits the maximum achievable accuracy, but in this stage of the project this is hardly a restriction, since nanometer level accuracy is not to be expected yet.

Chapter 3

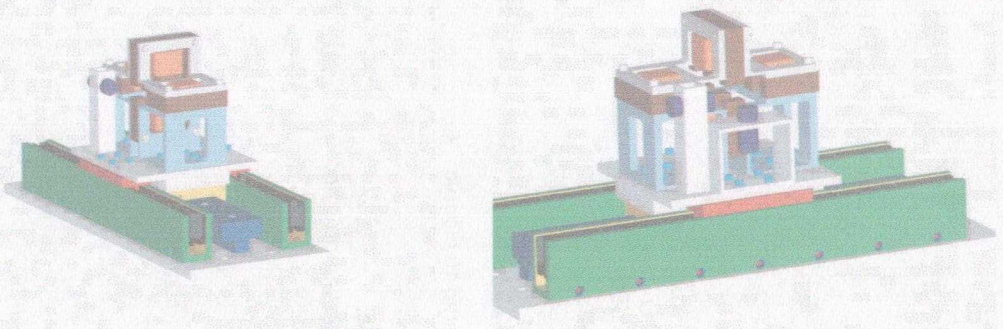
Linear motors

3.1 Hardware

In this chapter, the term linear motor describes a so-called 'moving unit' [10]. One moving unit consists of two BLMC92-A linear motors (see Figure 3.1). The motors slide through two parallel magnet tracks, and actuate a platform that floats on an air bearing. This platform carries the magnetic actuators that levitate the tray, and the capacitive sensors to measure position of the tray with respect to the moving unit.

The two linear motors are connected in parallel to one amplifier. Within the magnet track, each motor has only 1 translational DOF. Since the motors are mounted rigidly to the same platform, it is assumed their translation is equal. Feedback from only one motor is used for commutation. The position of the platform is measured with a laser system.

All cables to the moving unit are guided by an E-chain. This E-chain also carries the capacitive sensor wires, the cables that provide current to the magnetic actuators (coils), and the air supply for the air bearing. Pin-outs and other details about the electronic connections can be found in Appendix A.1.



(a) Front view

(b) Side view

Figure 3.1: One moving unit

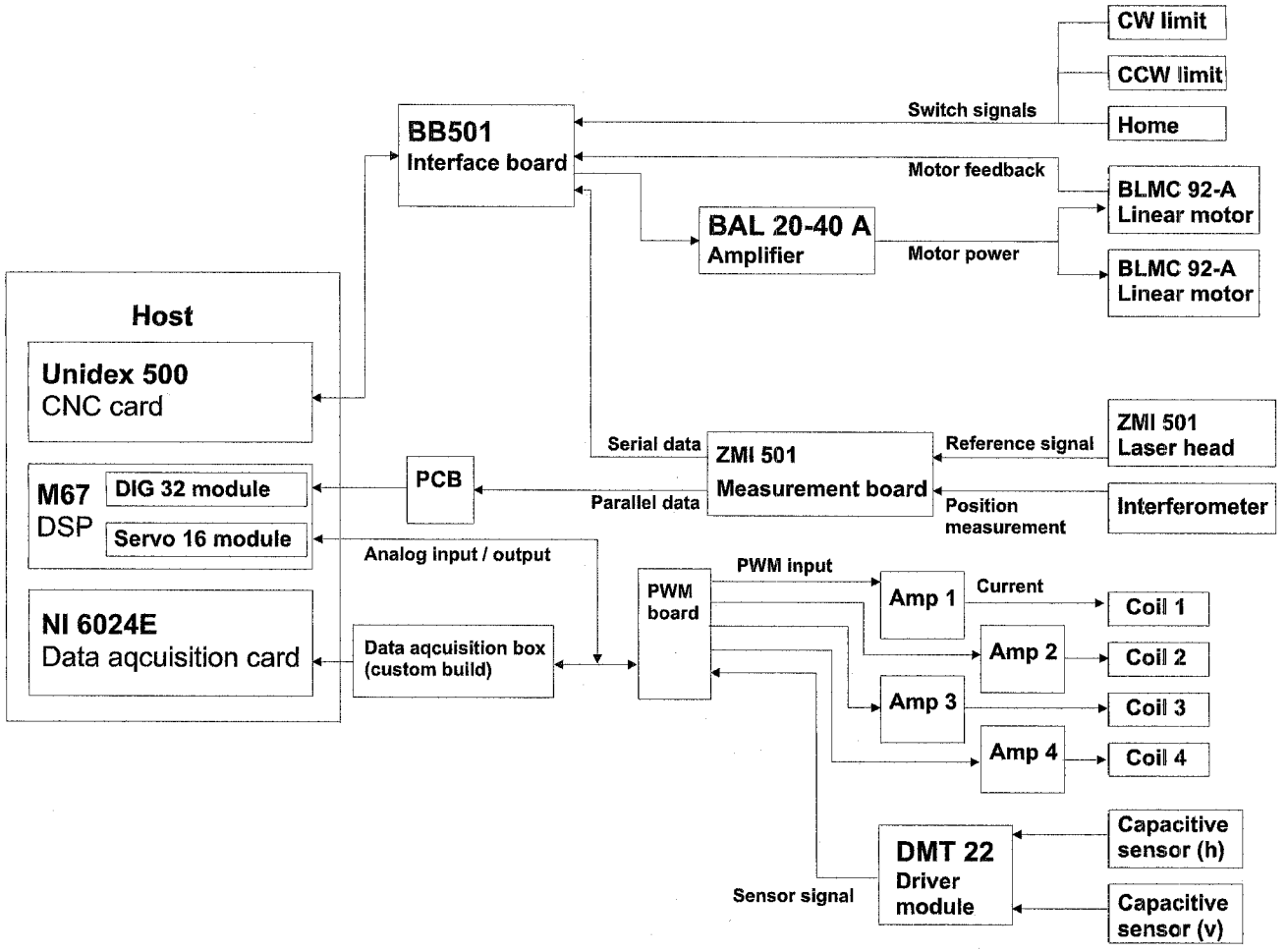


Figure 3.2: Schematic overview of one axis

3.2 Software

The position of the moving units is controlled by a Unidex 500 CNC-card and accessory MMI software. The U500 card gets input from a controller board (BB501), which receives position measurement from the laser system, feedback for commutation, status of limit and home switches, etc. A schematic overview of the system can be found in Figure 3.2. The structure of a PID-controller is preprogrammed in the MMI-software. After system identification and linear control design, the control gains can be set. More about this in Section 3.5.

If the software settings are correct, motion of the linear motors can be controlled with the CNC-card. An example program demonstrating some basic functionalities can be found in Appendix C.5. This program is written in so called G-code.

3.3 Laser measurement system

3.3.1 Operating principle

Translation of the moving units is measured using a Zygo ZMI 501 laser. This laser has a vacuum wavelength λ of 632.992 [nm]. The laser beam is split into two signals: a reference beam and a measurement beam. The measurement beam is reflected by a target mirror on the moving unit. Based on the phase difference between reference and reflected measurement beam, the displacement of the target is computed.

The laser has a maximum resolution of 1.24 [nm] ($=\frac{\lambda}{512}$). The controller in the MMI-software receives position feedback from the ZMI501 measurement board. This is serial data with a maximum rate of 23.04 [MHz], which limits the target velocity to 28.48 [mm/s]. When a lower laser resolution is used, a higher target velocity is possible. High accuracy also limits measurement range, but using the highest resolution a position range of approximately 1.33 [m] is possible. The maximum stroke per axis is approximately 0.208 [m], so this is no limitation.

The ZMI501 board also has parallel ports, which can be read using the digital I/O of the DSP. This functionality can be valuable later, when control loops for the linear motors and magnetic actuators have to be integrated for control of the 6-DOF system. An additional advantage is that now maximum resolution is possible without limitations on the target speed. Appendix C.1 contains C-code used for collecting parallel laser data with the DSP.

3.3.2 Layout laser system

Using several optical components, the laser signal can be split and directed to multiple target mirrors. The layout used to measure displacement on 4 axes, can be seen in figure 3.3. This layout was designed in [10], but is not the most efficient. Displacement on 4 axis could also be measured using less optical components.

In the original design of [10], no kinematic mounts for the optical components were included. These mounts can be used to secure the components, and provide freedom for small adjustments. These kinematic mounts are an absolute requisite for a good alignment.

In order to include kinematic mounts in the laser setup, the aluminum mounting blocks were redesigned. Drawings of the mounts and the modified aluminum blocks can be found in Appendix A.2.

3.3.3 Alignment of optical components

The alignment of the laser head, beam splitters, fold mirrors and interferometers is a time consuming proces that has to be performed accurately. For future reference, a step-by-step instruction is given in Appendix A.3.

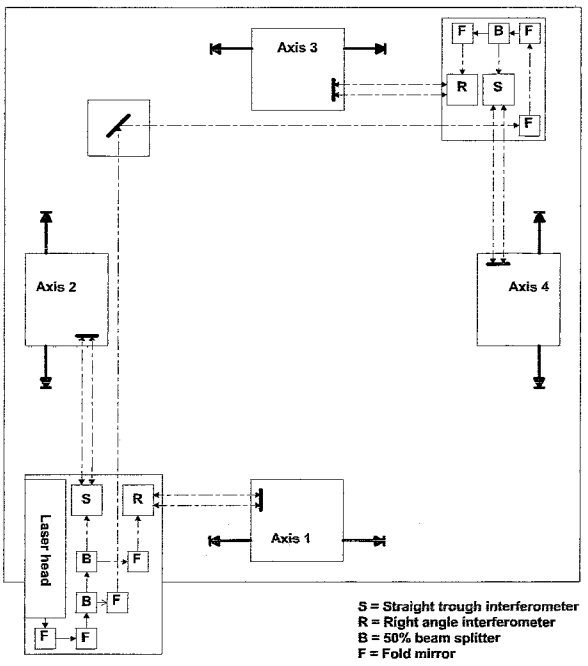


Figure 3.3: Laser setup

3.4 System identification

An external signal can be added to the control action of the preprogrammed controller on the U500. If all control gains are set to zero, the frequency response of a linear motor can thus be measured open loop by externally supplying white noise. The frequency response of the linear motor on axis 1 can be seen in Figure 3.4. White noise up to 400 [Hz] was injected.

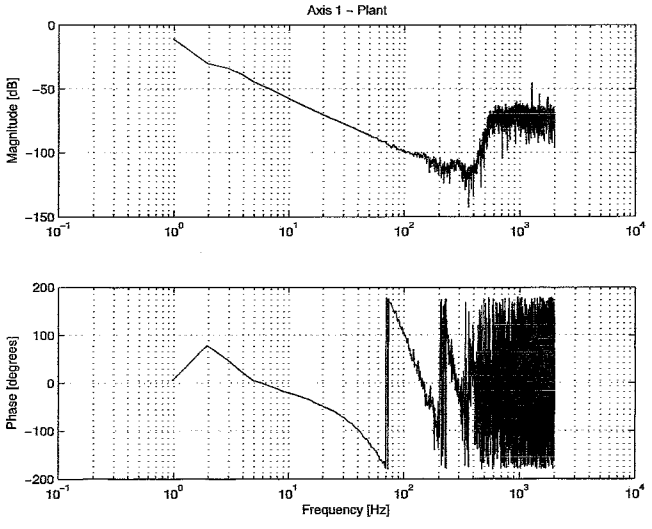


Figure 3.4: Frequency response of linear motor on axis 1

From this figure it can be concluded that the linear motor behaves as a mass. This was to be expected, since it floats on air bearings and experiences minimal friction. Only the stiffness of the E-chain (carrying all cables to the motor and coils) resists the translation of the motor, but this can be neglected.

The phase should be 0 or -180 degrees for a mass, but Figure 3.4 shows phase increases with frequency. This can be explained by the way this measurement was performed. The white noise is injected to the BB501-board, so it is sampled first. The delay of this A/D-conversion shows in the phase measurement. Also, position is measured with the laser system. A program running on the DSP converts the parallel output of the laser to a voltage, which is used for the frequency response measurement. The limited sample frequency of the program running on the DSP also introduces delay.

Theoretically, the first delay could be eliminated by measuring the frequency response between the 'echoed' white noise (white noise is sampled and sent to output channel, thus undergoing the delay of the A/D- and D/A-conversions) and the output voltage. In practice this is not possible because the BB501-board has no output channels available. The second delay can not be eliminated at all, since there is no other way to obtain a position signal using this setup.

Since a number of scaling factors is used in the MMI-software, a simple check is done to verify the results. The equation of motion for a single mass:

$$k_m u = m\ddot{x} \tag{3.1}$$

Here x [m] is motor translation, u [V] is torque command, m [kg] is the mass of the linear motor and k_m [N/V] is the motor constant. The ratio between mass and motor constant can be determined from Figure 3.4:

Frequency [Hz]	Magnitude [dB]	$\frac{k_m}{m}[\frac{N}{Vm}]$
10.00	-57.60	5.20
20.00	-70.40	4.77
30.00	-77.10	4.96
40.00	-82.20	4.90

Table 3.1: Ratio $\frac{k_m}{m}$ for several frequencies

The mean ratio $\frac{k_m}{m}$ equals 4.96 [N/Vm]. According to [1] and [4], the motor constant equals 18.96 [N/V], resulting in a mass of 3.82 [kg] for one moving unit. This is close to the estimated mass of 4.0 [kg], confirming that the result is correct.

Since the moving units on all 4 axes are identical, it is assumed this mass-model accurately describes them all.

3.5 Control

As mentioned before, the linear motors are controlled using the Unidex 500 CNC-card and accessory MMI-software. A fixed control structure is implemented in the software, so the user can only set the control gains. The structure of the controller is given in [3] (page 6-2), but this schedule contains several errors. The correct structure of the controller can be seen in Figure 3.5.

The sample frequency F_s is set to the maximum of 4000 [Hz]. The user can tune the gains K_{pos}, K_p, K_i and A_{ff}. The velocity feed forward gain V_{ff} is only used if a secondary feedback device is used to measure velocity. In this case velocity is derived from the position data, and V_{ff} should be set to 256. This results in a PID-controller with acceleration feed forward. Rewriting the scheme in Figure 3.5 shows how the gains of this PID-controller relate to K_{pos},K_p and K_i:

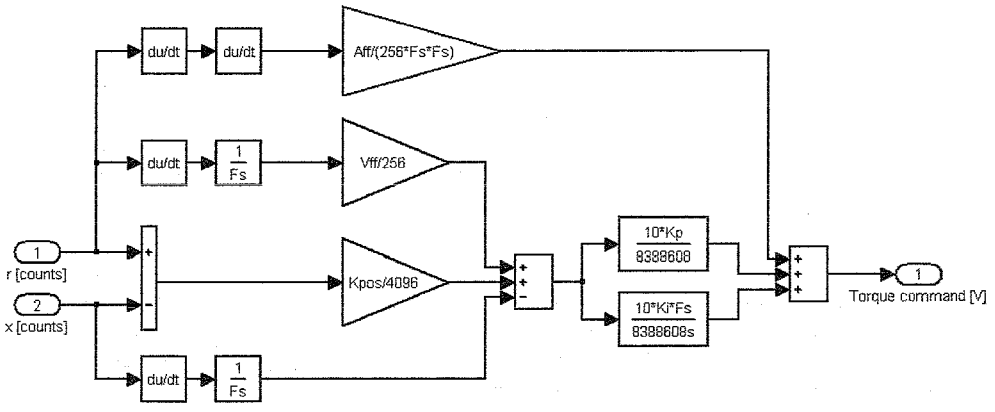


Figure 3.5: Controller as implemented in the MMI-software

$$P = \frac{K_{pos}}{4096}\frac{10K_p}{8388608} + \frac{10K_i}{8388608} \quad \text{[V/count]}$$

$$I = \frac{K_{pos}}{4096}\frac{10K_iF_s}{8388608} \quad \text{[V/s count]}$$

$$D = \frac{1}{F_s}\frac{10K_p}{8388608} \quad \text{[s V /count]}$$

This allows linear control design using DIET, after which the correct settings for K_{pos},K_p and K_i can be retrieved and implemented. Normally setpoint and position are measured in [m], in which case these gains have to be divided by the laser resolution [m/count].

The MMI-software has an option to automatically tune the controller gains, but manual tuning using DIET leads to better results. A second-order filter is preprogrammed as well, which can be used either as a notch filter or as a low-pass filter. When used as a notch, the resonance around 150 [Hz] can be reduced, at the cost of some phase loss leading to lower bandwidth. It is preferred to implement a low-pass filter to filter out high frequency noise. This filter has a double pole at 300 Hz.

Implementation of the feed-forward term led to 'strange' results. Consulting Aerotech Inc. [5] learned that the feed-forward implementation in the MMI-software contains errors, and that feed-forward should not be used until the next software upgrade becomes available. Consequently, the results shown here were obtained without feed-forward (A_{ff}=0).

3.6 Real-time results

Figure 3.6 shows that the closed loop bandwidth (45 degrees phase) lies between 50 and 55 [Hz]. The sensitivity (Figure 3.7) stays below 6 dB, which is a requirement for good disturbance rejection. Figure 3.8 shows that the steady state error $|e_{ss}| \leq 50$ [nm].

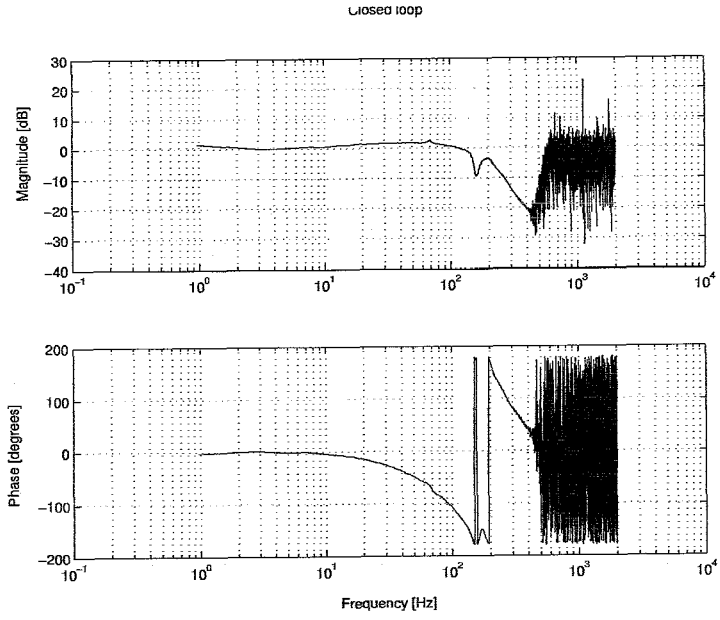


Figure 3.6: Axis 1: Closed loop measurement

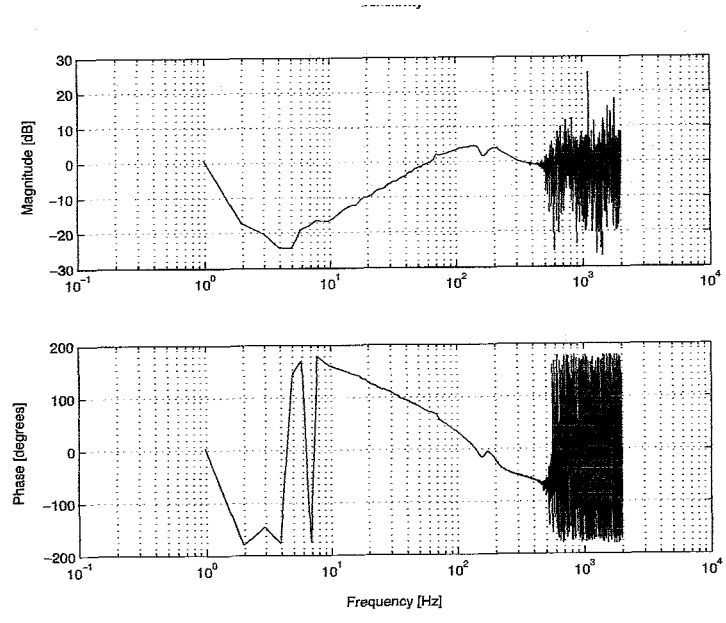


Figure 3.7: Axis 1: Sensitivity measurement

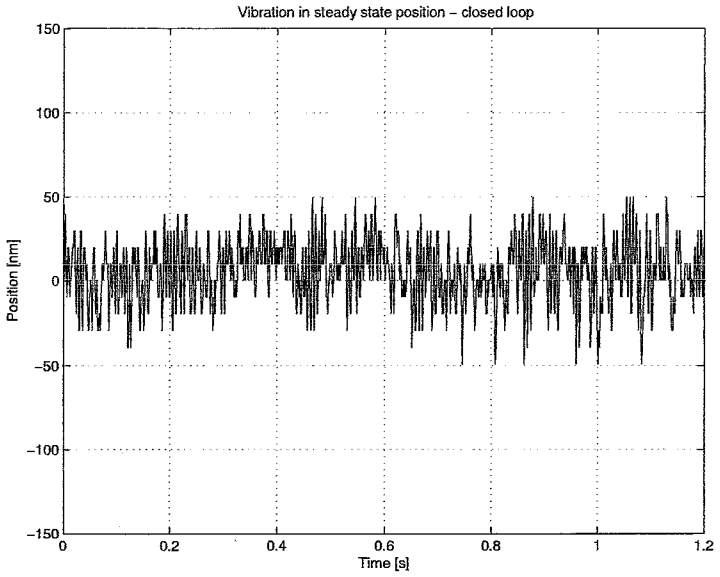


Figure 3.8: Steady state error, closed loop measurement

3.7 Conclusion

Hardware and software for the linear motor have been set up. This was a time consuming process; connecting all electronics, redesigning and aligning the laser system, ordering and replacing several devices, debugging software, etc. The implementation of a linear controller shows that this process has been completed successfully. Maximum achieved bandwidth is 50 [Hz], while steady state error is limited to 50 [nm].

Several limitations have been encountered, especially considering the restricted freedom in controller design. To improve performance, it will be necessary to implement a controller with more complex structure. For example, it is desired to implement a feed-forward term. Such a controller could be implemented on a DSP, and the control signal could be sent to the linear motors (bypassing the preprogrammed controller in the MMI-software). This controller could used parallel data instead of the serial data used now. At this moment such a solution is not possible, since the available DSP has 16 analogue output channels that are all needed to control the magnetic actuators.

Chapter 4

Force model of magnetic actuator

The magnetic actuator has been described previously in Chapter 2. In this chapter, a theoretical force model will be derived for the actuator. The model has been verified with measurements on an experimental setup.

This model will be used for nonlinear control of a rotating arm actuated by one magnetic actuator, and in the 6-DOF model of the total system.

In the electromagnetic theory in this chapter, the following definitions are used:

B	=	Flux density	[Wb/m^2]
F	=	Force	[N]
H	=	Magnetic field intensity	[A/m]
I	=	Current	[A]
N	=	Number of coil windings	[-]
R	=	Reluctance	[A/Wb]
w_m	=	Energy density	[J/m^3]
W_m	=	Energy	[J]
μ	=	Permeability of a material ($\mu = \mu_r \mu_0$)	[H/m]
μ_0	=	Permeability of free space, $4\pi \cdot 10^{-7}$	[H/m]
μ_r	=	Relative permeability of a material	[-]
Ψ	=	Flux	[Wb]

4.1 Theoretical model

4.1.1 Assumptions

The magnetic actuators in this machine behave like tractive type magnetic suspensions. A force model is derived using the electromagnetical theory as presented in [23] and especially [18].

The following assumptions are made:

- There are two closed flux paths (see Figure 2.5). Flux leakage is assumed negligible. That is, the flux paths only pass through the laminations and the air gaps between E- en I-laminations.
- The air gap x is small compared to the dimensions d_1, h_3 and h_4 (see Figure B.1).
- The metal core of the actuator does not saturate. This means the relation between B and H is linear in the operating region used.
- There are no Eddy current losses.

4.1.2 Model

As explained in Chapter 2, a magnetic field is generated when a current is passed through the coil around the center leg of the 'E'-shaped laminations. The work exerted by the magnetic force on the 'I'-laminations equals the change of magnetic energy stored in the air gap. This force is always directed so that it tends to reduce the air gap:

$$-Fdl = dW_m \tag{4.1}$$

In the linear operating range (no saturation), the flux density B is given by the permeability of the material times the field intensity H:

$$B = \mu H \tag{4.2}$$

The energy density is given by:

$$w_m = \frac{1}{2}BH = \frac{1}{2}\frac{B^2}{\mu} \tag{4.3}$$

Integrating over the volume gives the total energy:

$$W_m = \int w_m dV = S\int w_m dl = \frac{1}{2}\frac{B^2}{\mu}S\int dl \tag{4.4}$$

Now, substitution of Equation 4.4 in Equation 4.1 gives:

$$-Fdl = d[\frac{1}{2}\frac{B^2}{\mu}S\int dl] \tag{4.5}$$

This can be reduced to:

$$F = -\frac{1}{2}\frac{B^2}{\mu}S \tag{4.6}$$

The total flux through a surface is given by the integral of the flux density B over the surface:

$$\Psi_T = \int BdS \tag{4.7}$$

The flux generated by a current through the coil is given by:

$$\Psi_T = \frac{NI}{R} \tag{4.8}$$

Here, the reluctance R is defined as:

$$R = \frac{l}{\mu S} \tag{4.9}$$

Where l represents the magnetic path length. In Appendix B the total reluctance R_T of the actuator is derived. Distinction is made between the constant reluctance R_C of the actuator core, and the variable reluctance R_X of the (variable) air gap x.

$$R_T = R_C + R_X \tag{4.10}$$

Substituting R_T in Equation 4.8 gives the total generated flux. This flux passes through the cross section of actuator's center leg (S_1, see Figure B.1), so that Equation 4.7 equals:

$$\Psi_T = B_1 S_1 \tag{4.11}$$

Assuming no flux leakage occurs, the same flux returns through the two outer legs with cross section S_2:

$$\Psi_T = 2B_2 S_2 \tag{4.12}$$

The total tractive force on the 'I' is the sum of forces at all three legs of the 'E'. The force in the center leg is defined as F_1, the forces at the two outer legs as F_2 (they are equal due to symmetry). Using Equation 4.6, the total force can be described by:

$$F_T = F_1 + 2F_2 = -\frac{1}{2}\frac{B_1^2}{\mu_0}S_1 - \frac{B_2^2}{\mu_0}S_2 \tag{4.13}$$

This can be written as:

$$F_T = -\frac{N^2\mu_0}{2A}\frac{I^2}{(\frac{R_C}{2A\mu_r} + x)^2} \tag{4.14}$$

With:

$$A = \frac{S_1 + 2S_2}{2S_1S_2}$$

The numerical values for these parameters are:

A	=	$6.543 \cdot 10^3$	[1/m^2]
R_C	=	857.8	[A/Wb]
N	=	258	[-]
μ_0	=	$4\pi \cdot 10^{-7}$	[H/m]
μ_r	=	47000	[-]

4.2 Empirical model

4.2.1 Experimental set-up

Experimental data was gathered using the setup in which one arm has 1 rotational DOF (see Figure 4.1 and Figure 5.1). In the notation of [10] this arm is called a 'crossmember'. The 'I'-laminations are integrated in the crossmember, as described in Chapter 2. At both sides of the arm, 'E'-shaped laminations with a coil around the center leg are placed, that can only exert tractive forces. For the following experiments, only the upper coil is used.

A force sensor carries the weight of the crossmember and a known preload, generated by adding extra mass to the crossmember. Increasing the current through the upper coil results in an increasing actuator force in upward direction, while the resultant force is still directed downward. Since mass, preload, and all arms with respect to the point of rotation are known accurately, the actuator force can be derived from equilibrium of moments. The air gap is measured with a capacitive sensor.

By adjusting the height of the force sensor, the air gap between crossmember and actuator can be varied. For each air gap value, the current is increased stepwise until the actuator force exceeds the preload.

For several reasons, the magnitude of the applied preload is limited. It has to approximate a point force, and the force beam should not bend significantly under the preload. Since the preload is limited, only forces up to approximately 70 N can be measured. Higher actuator forces will pull the arm including weights up.

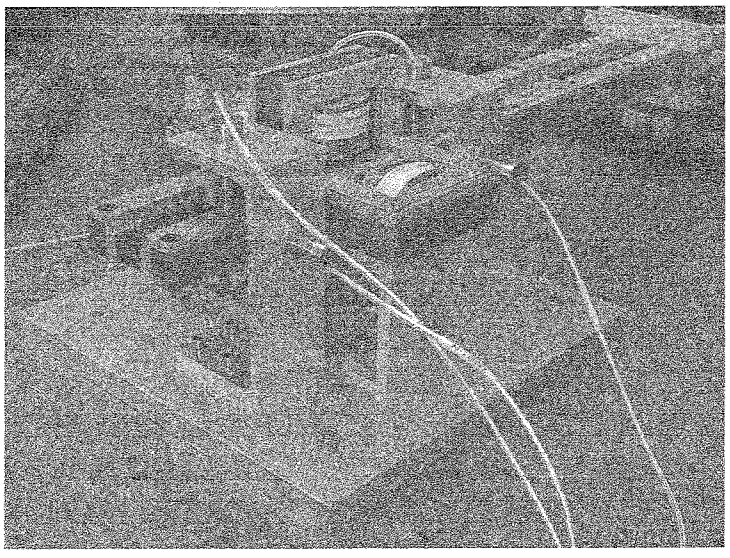


Figure 4.1: Arm with 1 rotational DOF in 1 magnet box

4.2.2 Results

Figure 4.2 shows the measured force as function of current and air gap. For good visualization, force has been plotted with a magnitude of 0 [N] for the region where the actuators force exceeds the preload (so the arm sticks to the upper actuator).

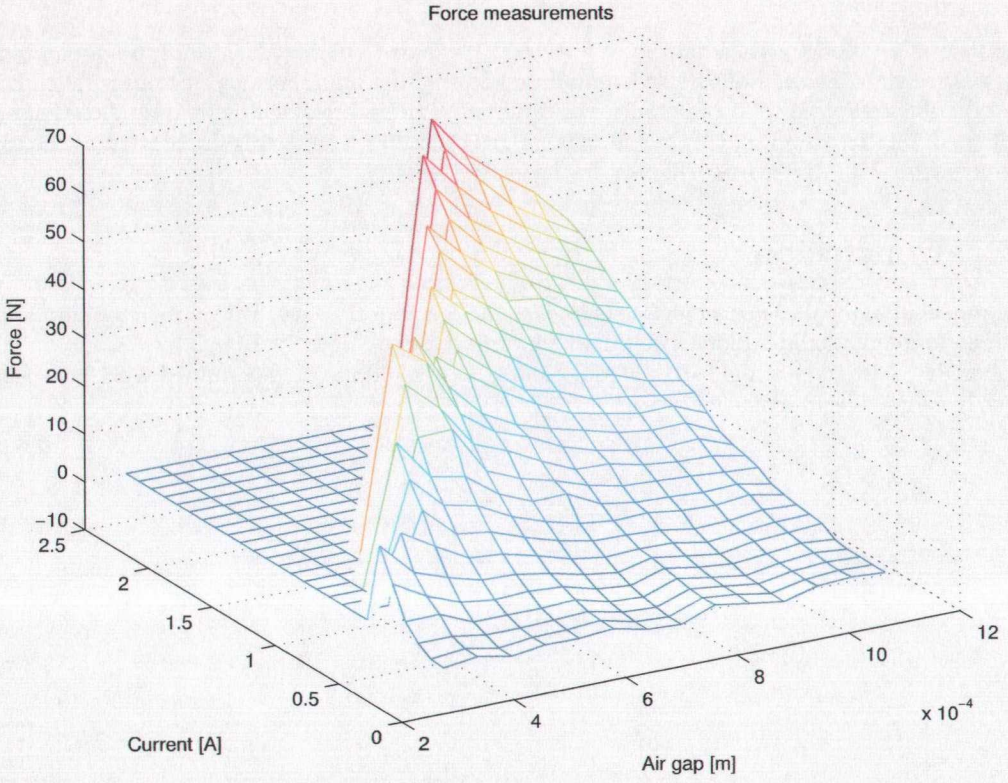


Figure 4.2: Force measurements

4.2.3 Model fit

In literature ([21],[7]) the following model is used to describe the tractive force in this type of magnetic suspension. Note that the structure is equal to 4.14:

$$F = K\frac{I^2}{(x_0 + x)^2}$$

With K a constant and x_0 an additional gap length to model the finite reluctance of the actuator's metal core. A Matlab-script (see Appendix D.1) was written to fit K and x_0 by minimizing the error criterium

$$E = \sum_1^i \sum_1^j |F_{meas}(i,j) - F_{mod}(i,j)|$$

With i and j the number of variations in current and position.

The best fit is achieved with the following values:

K	=	$7.5951 \cdot 10^{-6}$	[N m^2/A^2]
x_0	=	$73 \cdot 10^{-6}$	[m]

In Figure 4.3, the fitted model and measured data are plotted together. For a good visualization, the model is only plotted for those points where actual measurement were available. Figure 4.4 shows the difference between measurements and fit.

The theoretical model gives a gain K of $6.3824 \cdot 10^{-6}$ [N m^2/A^2], which is close to the empirically determined gain. The additional gap length found during measurements (73 micron) is much larger than the theoretical value (1.4 micron). The difference can be explained by the way the sensor was calibrated. The zero-level actually included an air gap caused by a protecting lip, preventing the cross-member from hitting actuator or sensor during experiments.

4.3 Conclusion

The model fit to the measurement data is close to the theoretical model. Due to practical limitations, no measurements could be done over the whole vertical range.

Possibly, saturation of the core material occurs. This could not be determined with the available tools, but should be investigated further.

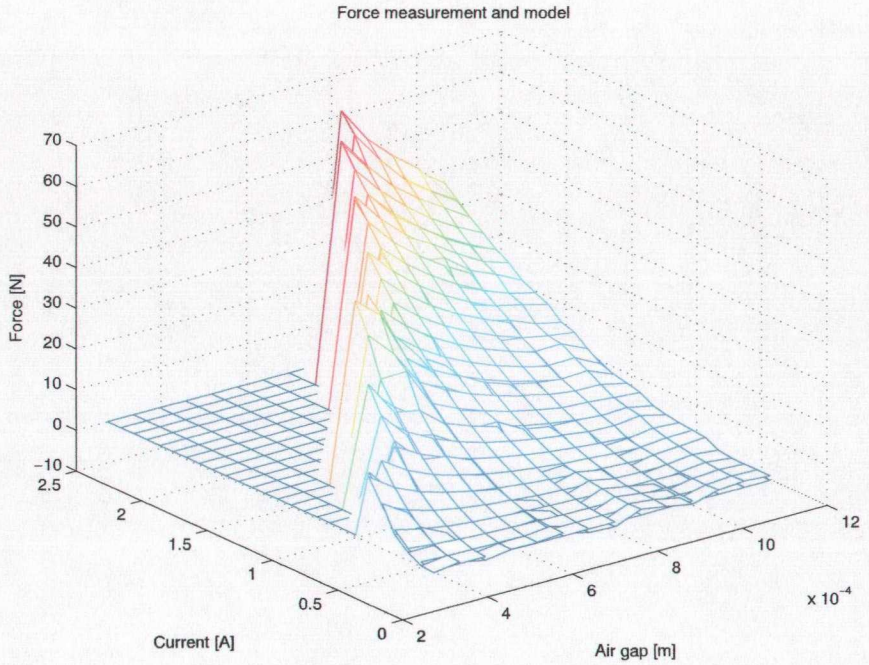


Figure 4.3: Force measurements and fit

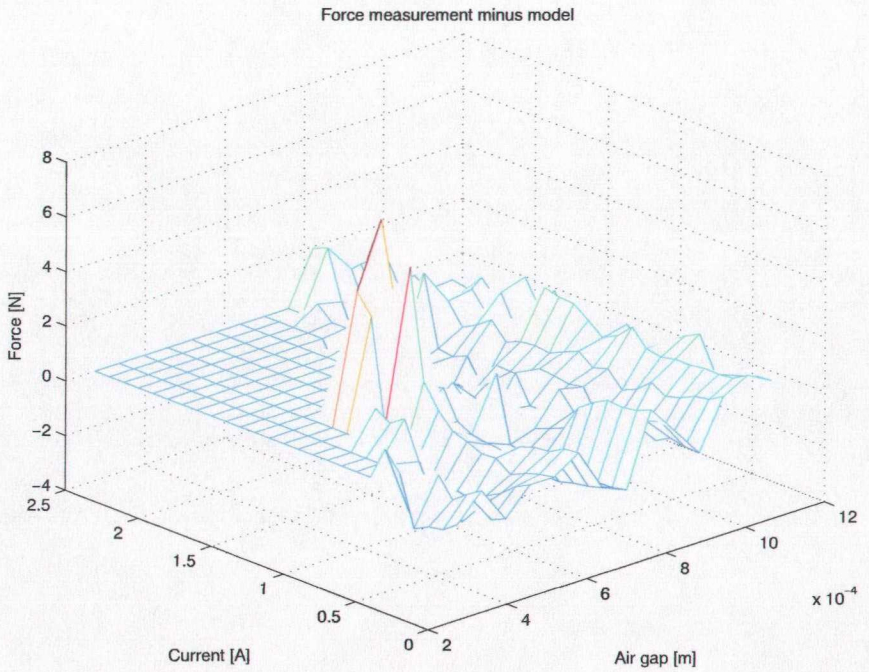


Figure 4.4: Difference between force measurements and fit

Chapter 5

Control of 1-DOF system with magnetic actuator

Before addressing the problem of controlling the 6-DOF system, a good understanding of the control of a magnetic actuator is necessary. To achieve this, a setup with a 1-DOF crossmember actuated by 1 magnetic actuator is used. Linear and non-linear controllers will be designed, implemented and compared using this system. Once a good control strategy is found, this can be extended for use in the more complex 6-DOF system.

In [10] several specifications are mentioned, which the 6-DOF system eventually has to meet. This concerns specifications like a maximum static error and the stroke for all DOF's. No explicit control specifications are defined in terms of bandwidth.

Since not all hardware is available, these specifications cannot be translated to specifications for the control of the 1-DOF setup. For example, in the 1-DOF setup position is measured with capacitive sensors. Their resolution is significantly lower than the resolution of the laser system. Also, the DSP used at this moment limits sample frequency of the controller and the amplifiers for the magnetic actuators have limited bandwidth.

In this stage of the project, the focus lies on finding a controller that is stable for the whole vertical stroke of the crossmember. Obviously, high bandwidth and small static error are desired. If a controller meeting these requirements is implemented with succes, it can be used as starting point for an iterative process to improve system performance. This proces will eventually have to lead to a competitive design. However, the necessary modifications of the system (position measurement with laser, magnetic suspension, faster DSP) will cost time and money and will not be performed before the end of this project.

More information about the implementation can be found in [22].

5.1 System description

The test set-up consists of a crossmember with 1 rotational DOF, which is actuated by a pair of opposite placed magnetic actuators. A schematic view of this system is shown in Figure 5.1. Using the force model derived in Chapter 4, the following equation of motion can be derived:

$$J_{xx}\ddot{\theta_1} = yK_1\frac{{I_1}^2}{{g_1(\theta_1)}^2} - yK_2\frac{{I_2}^2}{{g_2(\theta_1)}^2} - mgl\cos(\theta_1) \tag{5.1}$$

The description of the gaps:

$$g_1(\theta_1) = g_{nz} - y\tan(\theta_1)$$

$$g_2(\theta_1) = g_{nz} + y\tan(\theta_1)$$

With:

J_{xx}	=	Moment of inertia around x-axis	[kg m^2]
θ_1	=	Rotation around x-axis	[degrees]
m	=	Mass of crossmember	[kg]
g	=	Gravity constant	[m/s^2]
l	=	Distance from CM to point of rotation	[m]
y	=	Distance from magnetic actuators to point of rotation	[m]
K_1, K_2	=	Force constant	[N m^2/A^2]
I_1, I_2	=	Currents trough coil 1 and coil 2	[A]
g_{nz}	=	Nominal gap (when $\theta_1 = 0$)	[m]

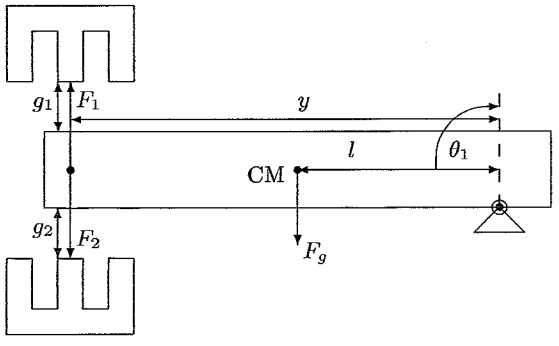


Figure 5.1: Schematic view of 1-DOF arm with one pair of actuators

5.2 Linear control

5.2.1 Linearization

In order to make the system single input, a premagnetization is introduced by applying bias currents to both coils. The variational current ΔI is the single control input:

$$I_1 = I_{b_1} + \Delta I \tag{5.2}$$

$$I_2 = I_{b_2} - \Delta I \tag{5.3}$$

Here I_{b_1} and I_{b_2} represent a bias current on coil 1 and coil 2, respectively. The control current has to meet the requirement $|\Delta I| \leq I_{b_1}, I_{b_2}$ to prevent the occurrence of 'negative' currents in both coils. Currents flowing in opposite direction will also generate tractive forces.

The state vector $\underline{x}$ consists of angle and angular velocity:

$$\underline{x} = \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \end{bmatrix} \tag{5.4}$$

$$\dot{\underline{x}} = \begin{bmatrix} \dot{\theta}_1 \\ \ddot{\theta}_1 \end{bmatrix} \tag{5.5}$$

$$\dot{\underline{x}} = \begin{bmatrix} x_2 \\ \frac{yK_1}{J_{xx}} \frac{(I_{b_1}+\Delta I)^2}{g_1^2(x_1)} \quad \frac{-yK_2}{J_{xx}} \frac{(I_{b_2}-\Delta I)^2}{g_2^2(x_1)} \quad \frac{-mgl\cos(x_1)}{J_{xx}} \end{bmatrix} \tag{5.6}$$

In order to linearize the system, define:

$$\begin{aligned} x_1 &= \overline{x}_1 + \delta x_1 \\ x_2 &= \overline{x}_2 + \delta x_2 \end{aligned} \tag{5.7}$$

Where δx denotes small perturbations with respect to the operating point $\overline{x}$. Now the system is linearized around the operating point ($\overline{x}_1 = 0, \overline{x}_2 = 0$) in the middle of the vertical range:

$$\begin{aligned} \delta\dot{x} &= \underline{A}\,\delta x + \underline{B}\,\Delta I \\ \delta y &= \underline{C}\,\delta x + \underline{D}\,\Delta I \end{aligned} \tag{5.8}$$

Taking the appropriate Jacobian and substituting the equilibrium point leads to the following linearized system :

$$\underline{A} = \begin{bmatrix} 0 & 1 \\ \frac{2y^2K_1I_{b_1}^2}{J_{xx}g_{nz}^3} + \frac{2y^2K_2I_{b_2}^2}{J_{xx}g_{nz}^3} & 0 \end{bmatrix} \tag{5.9}$$

$$\underline{B} = \begin{bmatrix} 0 \\ \frac{2yK_1I_{b_1}}{J_{xx}g_{nz}^2} + \frac{2yK_2I_{b_2}}{J_{xx}g_{nz}^2} \end{bmatrix} \tag{5.10}$$

$$\underline{C} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{5.11}$$

$$\underline{D} = \begin{bmatrix} 0 \end{bmatrix} \tag{5.12}$$

Assuming both actuators are equal ($K_1 = K_2$), the poles of the linearized system are:

$$\lambda = \pm\sqrt{\frac{2y^2K_1(I_{b_1}^2 + I_{b_2}^2)}{J_{xx}g_{nz}^3}} \tag{5.13}$$

The bias current I_{b_1} can be chosen 'arbitrarily' by the control designer. Typically, the bias currents equal half of the maximum current [7]. Since the current drive amplifiers saturate at 3.0 [A], $I_{b_1} = 1.5$ [A]. I_{b_2} follows from the value I_{b_1} and the condition that resultant force in the operating point is 0:

$$I_{b_2} = g_{nz}\sqrt{\frac{K_1}{K_2}\frac{{I_{b_1}}^2}{{g_{nz}}^2} - \frac{mgl}{yK_2}} \tag{5.14}$$

With a bias of $I_{b_1} = 1.5$ [A], $I_{b_2} = 1.02$ [A] and the poles are $\pm$ 283.2 [rad/s].

Actuator saturation

The amplifiers for the magnetic actuators are designed to deliver up to 4.0 [A], but only for short experiments. They will become extremely hot and will fail if maximum current is delivered for longer periods. Much heat development in the coils is also undesired. The amplifier gain was adjusted so that currents are limited to 3.0 [A], and the amplifiers can run continuously for long periods without problems.

5.2.2 System identification

To verify the linearization, frequency response of the system can be measured in the operating point. System identification for this system can only be performed closed loop. Once a linear controller is designed that stabilizes the system in the operating point ($\theta_1 = 0$), sensitivity can be measured. From the sensitivity measurement, open loop and subsequently plant can be derived. See Appendix D.2 for the M-file used.

Main challenge in this process is to design a stabilizing controller *before* the system identification is performed. In practice, this is an iterative process.

The plant derived from the sensitivity measurement can be seen in Figure 5.3. Figure 5.2 shows a Bode-plot of the linearized system equations. Note that there is a gain difference of roughly 80 [dB] between the two. This can be explained by the fact that in the sensitivity measurement, the sensor sensitivity ($50 \cdot 10^{-6}$ [m/V]) and arm y ($\frac{1}{0.335}$ [m]) are included, accounting for a 76 [dB] difference. Both plants are very similar considering gain and poles (at 45 Hz). In the real system, a resonance can be seen around 120 [Hz]. Note that the phase difference of 180 degrees represents a minus-sign. The real-time measurement shows additional phase delay at high frequencies, which is caused by the sampling of the white noise before it is injected to the system (as explained before in Chapter 3).

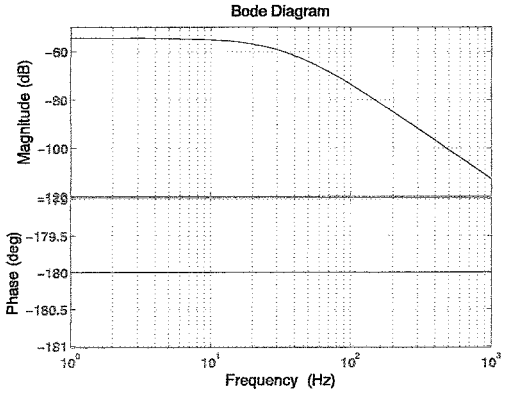


Figure 5.2: Bode plot of linearized system

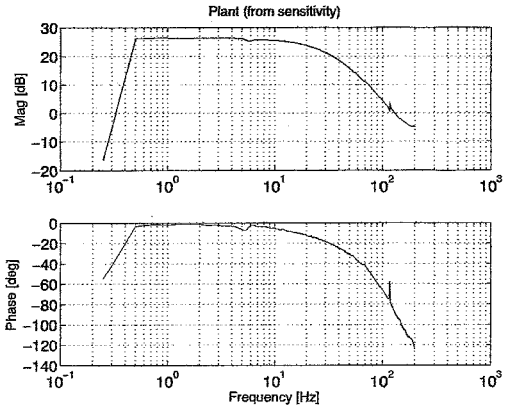


Figure 5.3: Plant derived from sensitivity measurement

5.2.3 Simulations

A linear controller for the linearized system was designed with DIET and analyzed in simulations. The simulated closed loop and sensitivity can be seen in Figure 5.4 and Figure 5.5. The bandwidth is approximately 55 [Hz]. Sensitivity stays below 6 [dB] for all frequencies.

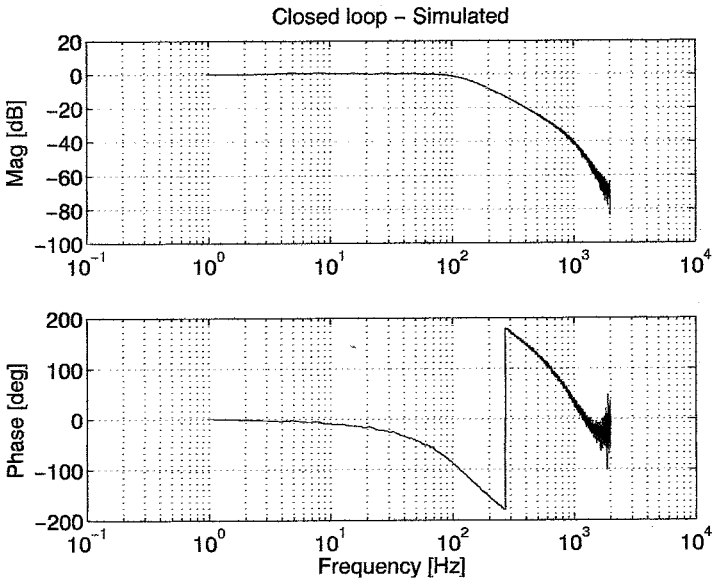


Figure 5.4: Linear control, closed loop (simulation)

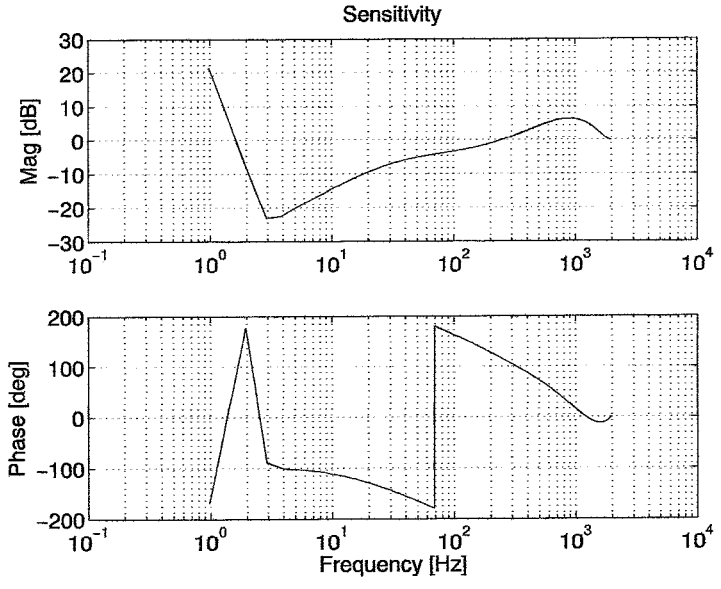


Figure 5.5: Linear control, sensitivity (simulation)

Since the linearized system equations are strictly valid for small variations around the equilibrium point, it is interesting to see how the system behaves for large strokes. Figure 5.6 shows the closed loop system can track a 5 [Hz] sinusoidal with amplitude $1.2 \cdot 10^{-3}$ [degrees], which is roughly 80% of the whole vertical range.

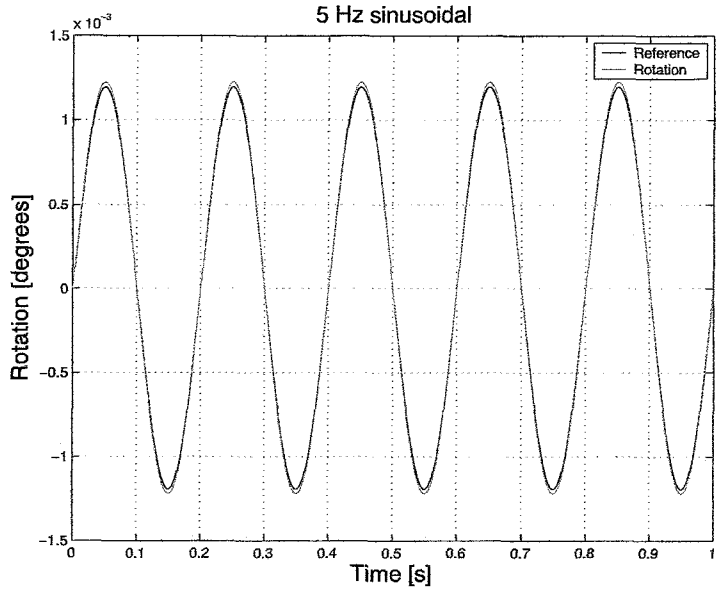


Figure 5.6: Tracking of a 5 Hz sinusoidal over large range (simulation)

5.2.4 Real time results

The implementation of a linear controller on this system is difficult. First problem is the fact, that a controller with good performance around the equilibrium position is not suited for the whole vertical range. Since the crossmember initially rests on the lower coil, it is far away from the operating point for which the controller is designed (the middle of the vertical range).

Second problem is that in the initial position the lower actuator exerts a theoretically infinite force on the crossmember as long as I_2 is non-zero. This means the crossmember is lifted when the control current ΔI equals the bias current I_{b_2}, so $I_2 = 0$. At this moment $I_1 = I_{b_1} + \Delta I$. It is not hard to see that this situation does not match the linearized system equations.

Because of these problems, implementation of the controller designed in simulation leads to an unstable system. A controller with lower gains was designed. The closed loop system can perform a 'smooth' step (acceleration is a linear function of time, so position is a third order curve) towards $\theta_1 = 0$. However, the performance of this controller is really poor, as can be seen in Figure 5.7 through Figure 5.17. While the single DOF of the system is a rotation, both reference, position and error are converted a displacement [m] at the position of the actuator. Position error in [m] at this point gives a good indication of the accuracy of the system.

The system can only track a low frequency sinusoidal with an amplitude of a few micron. A good closed loop frequency response measurement could not be performed.

5.2.5 Conclusion

The results obtained so far with linear control are poor. The system is highly nonlinear, so the linearized system equations only describe the system accurately close to the operating point in which it was linearized. As the crossmember moves away from the operating point, the quality of the linearization decreases and the performance of the system degrades. Since no magnetic suspension is available (yet), the crossmember initially rests on the lower actuator. In this position it is far from the operating point.

A linear controller with good performance, as tuned in simulation, could not be implemented. The system could only be stabilized with a linear controller with lower gains, but with very poor results. The steady state position error is approximately 1 micron.

In literature (for example [21]), similar experimental setups can be found in which the effect of gravity is cancelled out using hinges. The equilibrium position of the crossmember will be in the middle of the vertical range, not at the lower actuator. However, for this project it is not useful to modify the setup. First of all, the required time is not available. More important is the fact a controller is pursued that can position the crossmember accurately over the whole vertical range. Initially positioning the system in the equilibrium point might show that linear control works for small regions around that point, but the closed loop system will still not be stable for the whole vertical range. Final reason is that the 6-DOF system will experience the same problem, but in this case it cannot be solved with hinges.

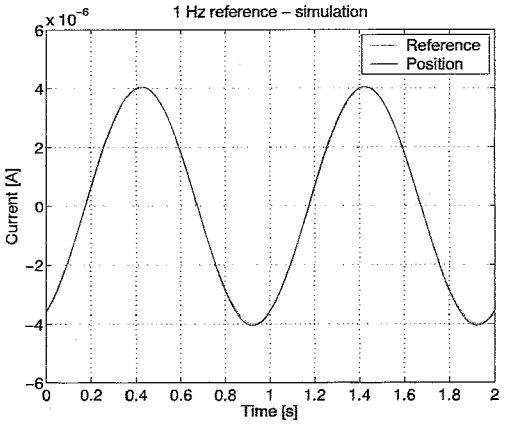


Figure 5.7: Reference and position (simulation)

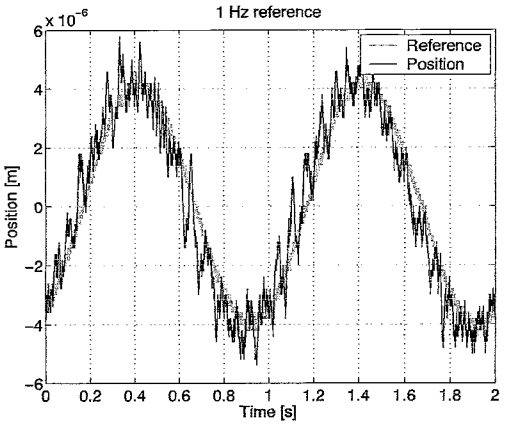


Figure 5.8: Reference and position (real time)

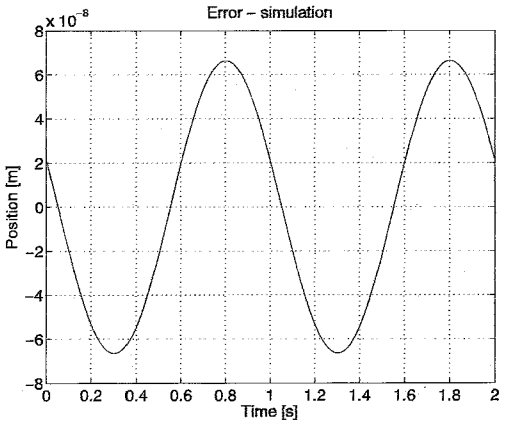


Figure 5.9: Error (simulation)

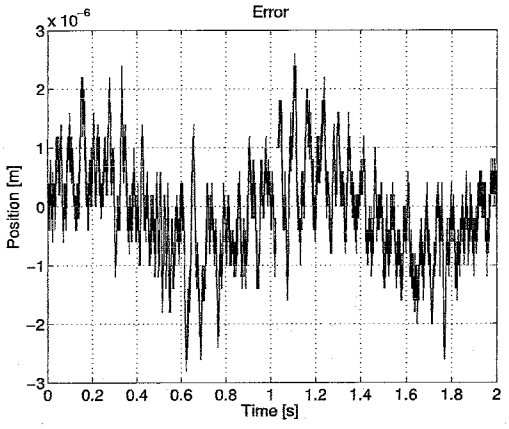


Figure 5.10: Error (real time)

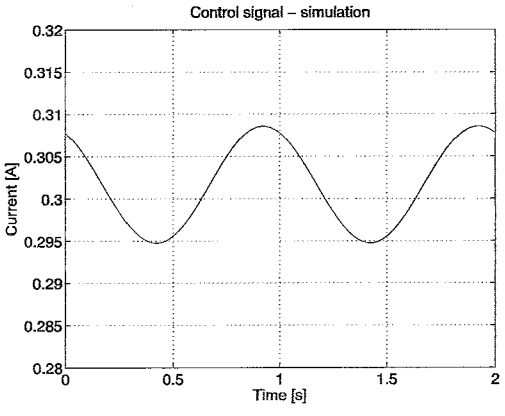


Figure 5.11: Control signal (simulated)

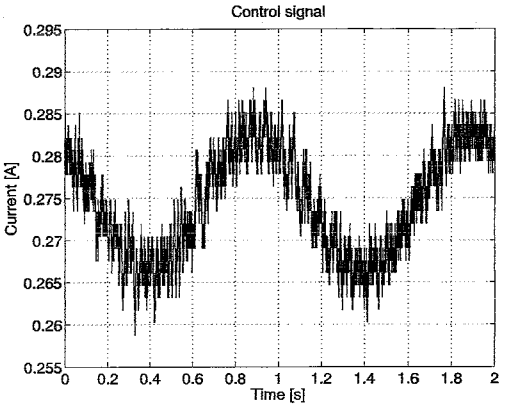


Figure 5.12: Control signal (real time)

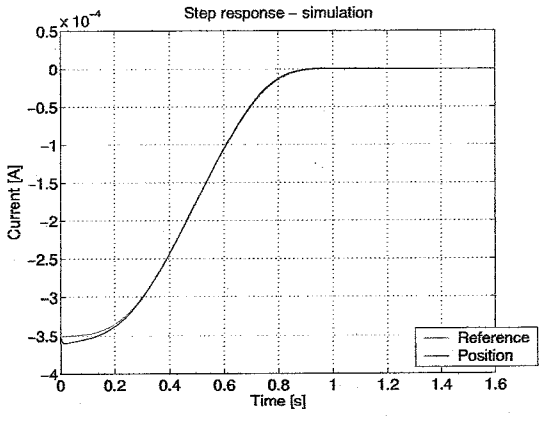


Figure 5.13: Reference and position (simulated)

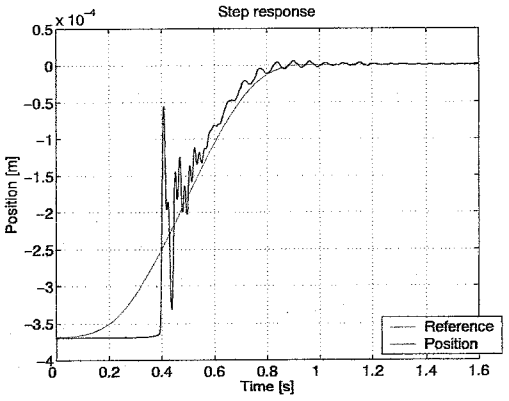


Figure 5.14: Reference and position (real time)

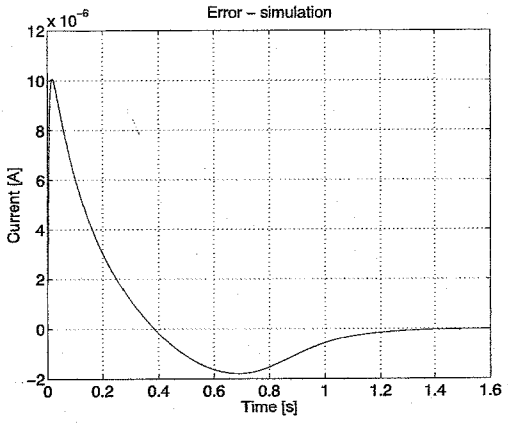


Figure 5.15: Error (simulated)

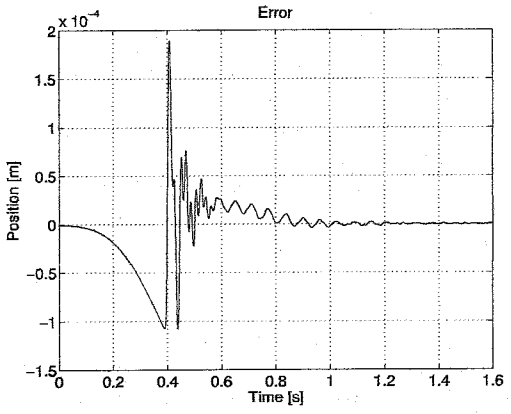


Figure 5.16: Error (real time)

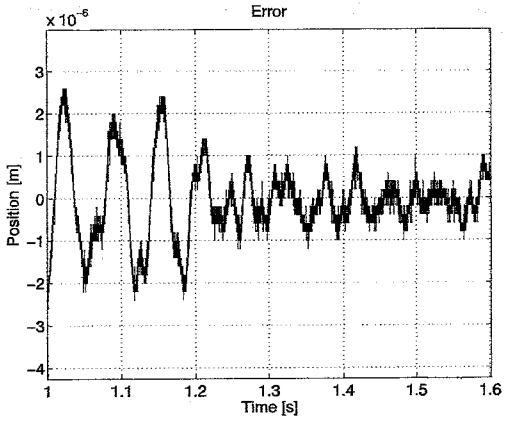


Figure 5.17: Steady state error

5.3 Non-linear control

The linear controller described in the previous section is only suited to control the system in a small range around the operating point. Literature ([21], [16], [17] and [15]) shows that consistent performance over the whole operating range can be obtained by feedback linearization of the system. This is described in the following subsection.

Theoretically, the feedback linearized system behaves as a double integrator and can be controlled with a standard linear controller. Drawback of the feedback linearization method is that it is not robust. In practice it will only work if the system model is exactly correct. Any model errors or unmodelled phenomena that introduce a mismatch between model and physical system will lead to deviations. Literature ([21], [17]) shows feedback linearization has successfully been implemented on similar magnetic suspension systems before.

Implementation of feedback linearization in combination with a linear controller leads to poor results. Oscillations occur when the crossmember is positioned close to the upper actuator. Although the force model derived in Chapter 4 closely matches empirical results, it can be concluded that this model is not 100% accurate. Differences between model and actual system can be the result of calibration errors, sensor drift, unmodelled friction, etc.

Since feedback linearized system could not be controlled satisfactory with a linear controller due to limited model accuracy, a different control strategy was used.

One option was the use of a robust controller. However, in general robust control leads to high order controllers, compromising fast implementation of the control algorithm. Even a simple controller like a PID could only be implemented at a maximum sample frequency of 4000 [Hz]. A high order robust controller would be very slow, especially for the 6-DOF system.

Sliding mode control is an other control strategy that provides robustness against modelling errors and unmodelled phenomena. Due to its relatively simple algorithm, it can be implemented with a high sample frequency. Sliding mode control is often used for this kind of suspension systems [7], but typically using voltage as input. Choosing voltage as input will make the system affine, but is not an option for this project. The amplifiers have been designed to drive current.

Two approaches to the sliding mode control of the system have been considered. In the first one, the system is made affine by applying feedback linearization. Sliding mode control of a linear system is applied according the theory of [14],[20] and [8], using force as input. Theory of this approach is described in subsection 5.3.2.

Subsection 5.3.3 describes the second approach, where the system is considered non-affine using currents as input and the theory of [12] and [13] is applied.

Both approaches were compared in simulation, with very similar results. Since very limited experimental time was left, only the first approach has been implemented and optimally tuned. Further investigation of the second approach deserves attention.

5.3.1 Feedback linearization

To obtain consistent performance over the whole operating range, feedback linearization was applied. Here, a nonlinear state transformation is described that cancels nonlinearities, so that the system dynamics are reduced to a linear form (double integrator).

Goal of the feedback linearization is to obtain the simple relationship

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= v \end{aligned} \tag{5.15}$$

In this equation v represents the angular acceleration to be applied to the crossmember. From Equation 5.6 it follows that:

$$v = \frac{yK_1}{J_{xx}} \frac{I_1^2}{g_1^2(x_1)} \quad \frac{-yK_2}{J_{xx}} \frac{I_2^2}{g_2^2(x_1)} \quad \frac{-mgl\cos(x_1)}{J_{xx}} \tag{5.16}$$

The following transformation linearizes the system in terms of the new input v:

$$\begin{aligned} I_1 &= g_1(x_1)\sqrt{\tfrac{J_{xx}v - mgl\cos(x_1)}{yK_1}} \quad \text{and} \quad I_2 = 0 \quad \text{for} \quad (J_{xx}v - mgl > 0) \\ I_2 &= g_2(x_1)\sqrt{\tfrac{-J_{xx}v + mgl\cos(x_1)}{yK_2}} \quad \text{and} \quad I_1 = 0 \quad \text{for} \quad (J_{xx}v - mgl < 0) \end{aligned} \tag{5.17}$$

Note that, using this transformation, at any instant current is flowing in only one of the two coils. Desired resultant forces can also be achieved with two actuators active at the same time, but the above result is simple and most efficient considering energy consumption.

5.3.2 Sliding mode control of feedback linearized system

The system including feedback linearization can be written in the standard state space formulation $\dot{x} = Ax + Bu$:

$$\begin{bmatrix} \dot{\theta}_1 \\ \ddot{\theta}_1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{21} \end{bmatrix} \begin{bmatrix} u \end{bmatrix} \qquad \text{with } \theta_1(0) = \theta_0 \tag{5.18}$$

With $x \in \mathbb{R}^n$, the $u \in \mathbb{R}^m$ and $\text{rank}(B) = m$. Now, the desired state vector is defined as:

$$x_d(t) = \begin{bmatrix} \theta_{1_d}(t) \\ \dot{\theta}_{1_d}(t) \end{bmatrix} \tag{5.19}$$

Assume that the desired state can be realized, with a bounded input signal, so that for $t > 0$:

$$\dot{x}_d = Ax_d + Bu_d \tag{5.20}$$

The tracking error is defined as $e = (\theta_1 - \theta_{1_d})$, so:

$$\dot{e} = Ae + B(u - u_d) \qquad \text{with } e(0) = \theta_0 - \theta_{1_d}(0) \tag{5.21}$$

Writing $B = [0 \quad H]^T$ with $H \in \mathbb{R}^{mxm}$ leads to:

$$\dot{e_1} = a_{11}e_1 + a_{12}e_2 \tag{5.22}$$

$$\dot{e_2} = a_{21}e_1 + a_{22}e_2 + H(u - u_d) \tag{5.23}$$

A sliding surface S with integral term is chosen:

$$S = \dot{e} + 2\lambda e + \lambda^2 \int_0^t e\, dt \tag{5.24}$$

The derivative of S is:

$$\dot{S} = \dot{e}_2 + 2\lambda\dot{e}_1 + \lambda^2 e_1 \tag{5.25}$$

Substitution of 5.22 and 5.23 gives:

$$\dot{S} = (a_{21} + 2\lambda a_{11} + \lambda^2)e_1 + (a_{22} + 2\lambda a_{12})e_2 + \hat{H}(u - u_d) \tag{5.26}$$

Define $A_{11} = (a_{21} + 2\lambda a_{11} + \lambda^2)$ and $A_{21} = (a_{22} + 2\lambda a_{12})$:

$$\dot{S} = A_{11}e_1 + A_{21}e_2 + \hat{H}(u - u_d) \tag{5.27}$$

Now, the following control law is chosen:

$$u = u_d - \hat{H}^{-1}(u_s + A_{11}e_1 + A_{21}e_2) \tag{5.28}$$

It is easy to see that substitution leads to $\dot{S} = -u_s$. An appropriate choice for input u_s, like $u_s = g_k \text{ sign}(S)$, will force S to 0. This can be verified by analyzing the Lyapunov-function $V = \frac{1}{2}S^2$. Since the sign-term leads to a discontinuous control action (chattering), it is replaced by the continuous term $u_s = g_k \text{ atan}(\phi^{-1}S)$. Implementation of this control action will force S to stay within a boundary layer with width 2ϕ.

For the feedback linearized system, $a_{11} = a_{21} = a_{22} = b_{11} = 0$ and $a_{12} = b_{21} = 1$. Accordingly, $\hat{H} = 1$, $A_{11} = \lambda^2$ and $A_{21} = 2\lambda$. The control law simplifies to:

$$u = u_d - (u_s + \lambda^2 e_1 + 2\lambda e_2) \tag{5.29}$$

Note that input u equals the desired angular acceleration v of the crossmember. The input currents I_1 and I_2 are computed using the nonlinear transformation described in Equation 5.17.

5.3.3 Sliding mode control of non-affine system with algebraic input invertibility

Here, the theory of [12] and [13] is applied to the system. The system equations, defined earlier in Equation 5.6, can be written as:

$$\dot{x} = f(x) + Bh(I_1, I_2, x) \tag{5.30}$$

With:

$$f_1(x) = x_2 \tag{5.31}$$

$$f_2(x) = -\frac{mgl\cos(x_1)}{J_{xx}} \tag{5.32}$$

$$B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{5.33}$$

$$h(I_1, I_2, x) = \frac{yK_1}{J_{xx}}\frac{I_1^2}{g_1^2(x_1)} - \frac{yK_2}{J_{xx}}\frac{I_2^2}{g_2^2(x_1)} \tag{5.34}$$

Switching

In [12], switching term and equivalent control are derived separately. To determine the switching term, first the sliding surface is defined:

$$S(x,t) = \dot{e} + 2\lambda e + \lambda^2 \int_0^t e\,dt \quad \text{with} \quad e = (x - r) \tag{5.35}$$

Asymptotic stability is guaranteed if the following condition is met:

$$S^T\frac{dS}{dt} = S^T(\frac{\delta S}{\delta x}f + \frac{\delta S}{\delta x}Bh(I_1, I_2, x) + \frac{\delta S}{\delta t}) < 0 \tag{5.36}$$

In case $S > 0$, the sliding condition is $\frac{dS}{dt} < 0$. Define:

$$\rho(x) = (\frac{\delta S}{\delta x}f + \frac{\delta S}{\delta t}) \tag{5.37}$$

Now the sliding condition is: $\rho(x) + h(I_1, I_2, x) < 0$. Choose $n(x)$ such that it is a upper boundary for $|\rho(x)|$:

$$0 \leq |\rho(x)| < n(x) \tag{5.38}$$

Least control effort is needed when only 1 coil current is used to meet the sliding condition. The following switching control term always meets the sliding condition for $S > 0$:

$$I_2 = g_2(x_1)\sqrt{\frac{J_{xx}n(x)}{yK_{2_{min}}}} \qquad \text{and} \qquad I_1 = 0 \tag{5.39}$$

With $K_{2_{min}}$ a lower boundary for K_2.

Note that the sliding condition is already met when $S > 0$ and $\rho(x) < 0$, but the input proposed above will be given anyway. Later more about this.

In a similar way, a switching control action for $S < 0$ can be derived:

$$I_1 = g_1(x_1)\sqrt{\frac{J_{xx}n(x)}{yK_{1_{min}}}} \qquad \text{and} \qquad I_2 = 0 \tag{5.40}$$

Equivalent control

The equivalent control is the control when the system is in sliding mode:

$$\dot{S} = \frac{\delta S}{\delta t} + \frac{\delta S}{\delta x}\dot{x} = \frac{\delta S}{\delta t} + \frac{\delta S}{\delta x}(\begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} h(I_1, I_2, x)) = 0$$

It follows that:

$$h_{eq} = -\frac{\delta S}{\delta t} - 2\lambda f_1(x) - f_2(x) = -\rho(x)$$

Solving for I_{eq} gives:

$$I_{2_{eq}} = g_2(x_1)\sqrt{\frac{J_{xx}\rho(x)}{yK_2}} \quad \text{and} \quad I_{1_{eq}} = 0 \quad \text{if} \quad \rho(x) > 0$$

$$I_{1_{eq}} = g_1(x_1)\sqrt{\frac{J_{xx}\rho(x)}{yK_1}} \quad \text{and} \quad I_{2_{eq}} = 0 \quad \text{if} \quad \rho(x) < 0$$

Stability

In the theory of [12], which is applied here, it is assumed that the equivalent control and switching term can be added directly. This can lead to the case where both actuators are active at the same moment, which does not seem logical (considering energy consumption, for example). Stability can be proved for all possible situations:

$$I = I_{sw} + I_{eq} \tag{5.41}$$

Stability can be proved for all possible situations:

1. $S > 0$ and $\rho > 0$
2. $S > 0$ and $\rho < 0$
3. $S < 0$ and $\rho > 0$
4. $S < 0$ and $\rho < 0$

For case 1: $I_1 = 0$ and $I_2 = g_2(x_1)\sqrt{\frac{J_{xx}\rho(x)}{yK_2}} + g_2(x_1)\sqrt{\frac{J_{xx}n(x)}{yK_{2_{min}}}}$. Stability is guaranteed if:

$$(\rho(x) + h) < 0 \tag{5.42}$$

Substituting I_1 and I_2 in h and simplifying gives:

$$h = -\left[\rho(x) + \frac{2yK_2}{J_{xx}}\sqrt{\frac{J_{xx}\rho(x)}{yK_2}}\sqrt{\frac{J_{xx}n(x)}{yK_{2_{min}}}} + \frac{K_2 n(x)}{K_{2_{min}}}\right] \tag{5.43}$$

If Equation 5.43 is substituted in Equation 5.42, the term $\rho(x)$ cancels out. Since all parameters are positive, both resulting terms are negative and stability is guaranteed:

$$-\frac{2yK_2}{J_{xx}}\sqrt{\frac{J_{xx}\rho(x)}{yK_2}}\sqrt{\frac{J_{xx}n(x)}{yK_{2_{min}}}} - \frac{K_2 n(x)}{K_{2_{min}}} < 0 \tag{5.44}$$

For case 2, $I_1 = g_1(x_1)\sqrt{\frac{J_{xx}-\rho(x)}{yK_1}}$ and $I_2 = g_2(x_1)\sqrt{\frac{J_{xx}n(x)}{yK_{2_{min}}}}$. Substituting I_1 and I_2 in h and simplifying gives:

$$h = -\rho(x) - \frac{K_2 n(x)}{K_{2_{min}}} \tag{5.45}$$

So:

$$(\rho(x) + h) = -\frac{K_2 n(x)}{K_{2_{min}}} \tag{5.46}$$

Since K_2, $K_{2_{min}}$ and $n(x)$ are positive, the sliding condition is met. For case 3 and 4 stability can be guaranteed in a similar way.

As mentioned before, this approach has not been implemented real-time due to time constraints. In future research, this approach could be investigated further, since it has lead to good results on a setup similar to the one used in this project [12].

5.3.4 Real time results

Results of feedback linearization in combination with sliding mode control (first approach) can be seen in Figure 5.18 through Figure 5.27.

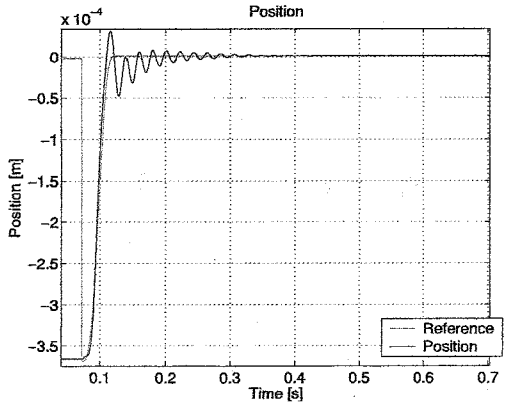


Figure 5.18: Smooth step, position and reference

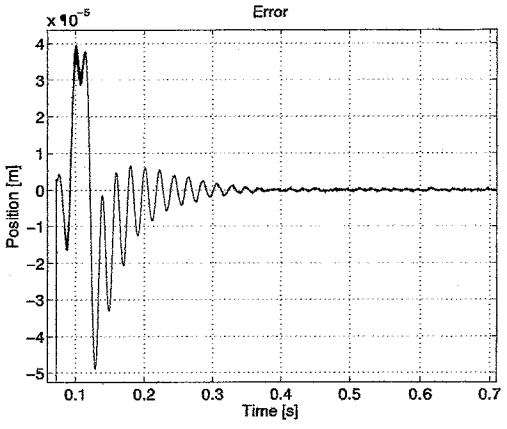


Figure 5.19: Smooth step, error

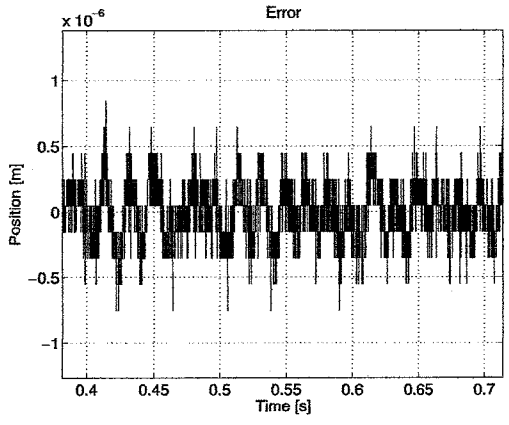


Figure 5.20: Smooth step, error (zoomed)

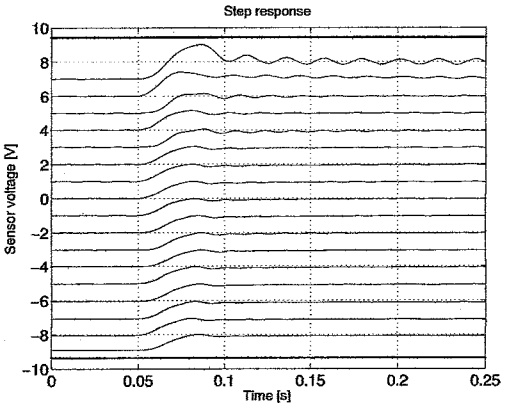


Figure 5.21: Steps of 50 μm

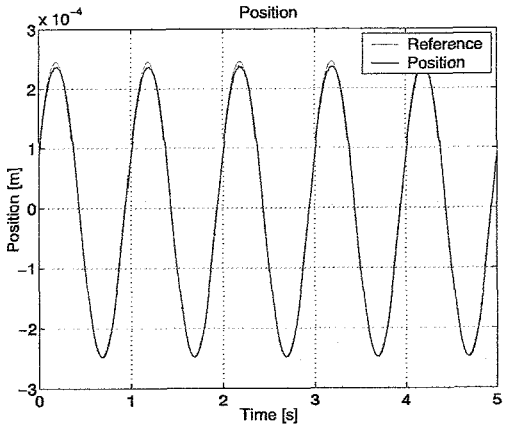


Figure 5.22: 1 Hz reference, amplitude 250 μm.

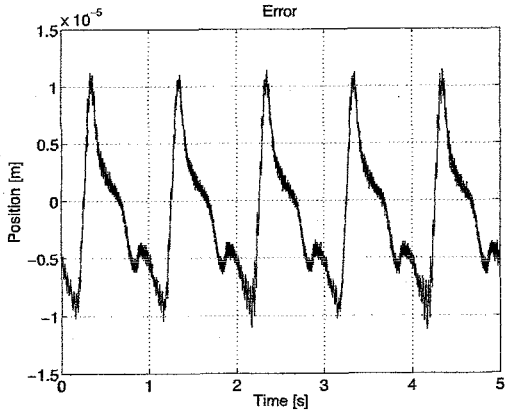


Figure 5.23: 1 Hz reference, amplitude 250 μm.

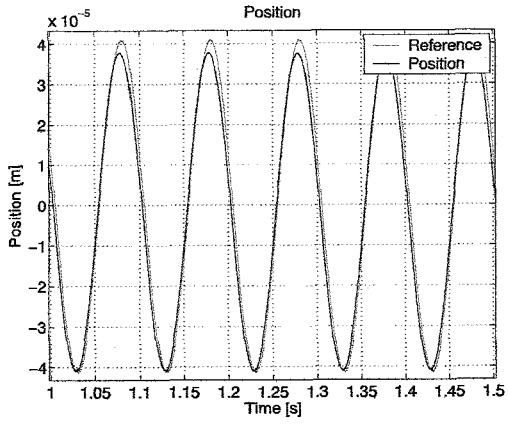


Figure 5.24: 10 Hz reference, amplitude 40 μm.

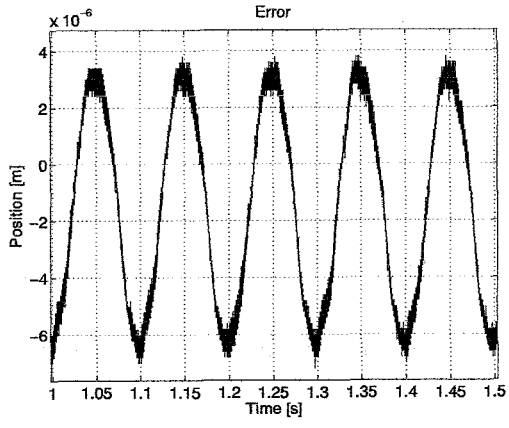


Figure 5.25: 10 Hz reference, amplitude 40 μm.

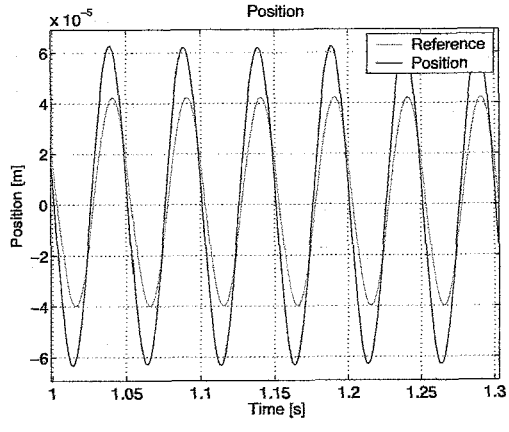


Figure 5.26: 20 Hz reference, amplitude 40 μm.

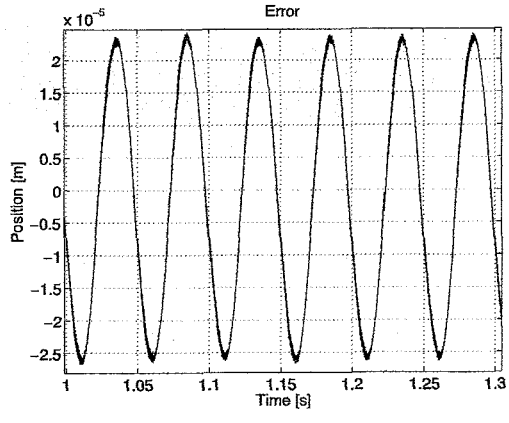


Figure 5.27: 20 Hz reference, amplitude 40 μm.

5.3.5 Conclusion

From the real time results, it can be concluded that a sliding mode controller in combination with feedback linearization has a larger operating range than a linear controller. Figure 5.21 shows steps of 50 [μm] each over the whole vertical range. Clearly, oscillations occur when the crossmember comes close to the upper actuator. If a setpoint closer than approximately 75 [μm] from the upper actuator is given, the crossmember sticks to this actuator. Apparently the force model of the actuator is not exact for this region, which falls within the region where no force measurement could be performed (see Chapter 4).

As Figure 5.22 and Figure 5.23 show, the system can track a 1 [Hz] sinusoidal with amplitude 250 [μm], which corresponds to half of the vertical range. Tracking error rapidly increases when when frequency is increased (see Figure 5.24 through Figure 5.27), which indicates that the relation between air gap, current and actuator force is frequency dependent.

Figure 5.18 through Figure 5.20 show tracking of a 'smooth' step from initial position to half of the vertical range. The steady state error lies well below 1 [μm].

Chapter 6

Modelling and control of the 6-DOF system

Due to the previously mentioned problems with the amplifiers, the project was considerably delayed. In the meanwhile, some preliminary work was done on the modelling and control of the 6-DOF system. The work presented in this chapter should be seen as a starting point for future research. A 6-DOF model is derived based on results of system identification of the linear motors and the force model derived in Chapter 4. An approach is suggested to distribute the control action for 6 DOF's over 20 actuators. Preliminary simulation results are shown.

6.1 System model

This section describes the modelling of the MIMO-system. Definitions can be found in Figure 6.1, Figure 6.2 and Figure 6.3.

6.1.1 Assumptions and definitions

The following assumptions were made while modelling the system:

- Only rigid body modes are considered. The lowest eigenfrequency is a flexible mode of the tray in vertical direction, and lies at approximately 74 [Hz]. For more details see [10].
- The linear motors only have 1 translational DOF each: x_1 and x_2 for motors 1 and 3, and y_1 and y_2 for motors 2 and 4. Their air bearings provide infinite stiffness in z-direction. The motors behave as a pure mass.
- Height and thickness of the arms are neglected in the computation of gaps, moments and forces.
- There are two types of actuators in the system: linear motors and magnetic actuators. Forces generated by the linear motors are described by:

$$F_{m_i} = k_i V_i \tag{6.1}$$

with k_i the motor constant and V_i the torque command in [V]. Index $i = 1...4$ gives the number of the linear motor. The forces generated by the magnetic actuators are modelled as:

$$F_{ac} = \frac{k_{ac} I_{ac}^2}{(g_0 + g_{ac})^2} \tag{6.2}$$

with parameter values as derived in Chapter 4. Here, I is the current input [A], k is a force constant $[\frac{Nm^2}{A^2}]$ and g is the air gap [m]. The magnet box on each axis (index $a = 1...4$) consists of 4 coils (index $c = 1...4$). See Figure 6.2 for definitions.

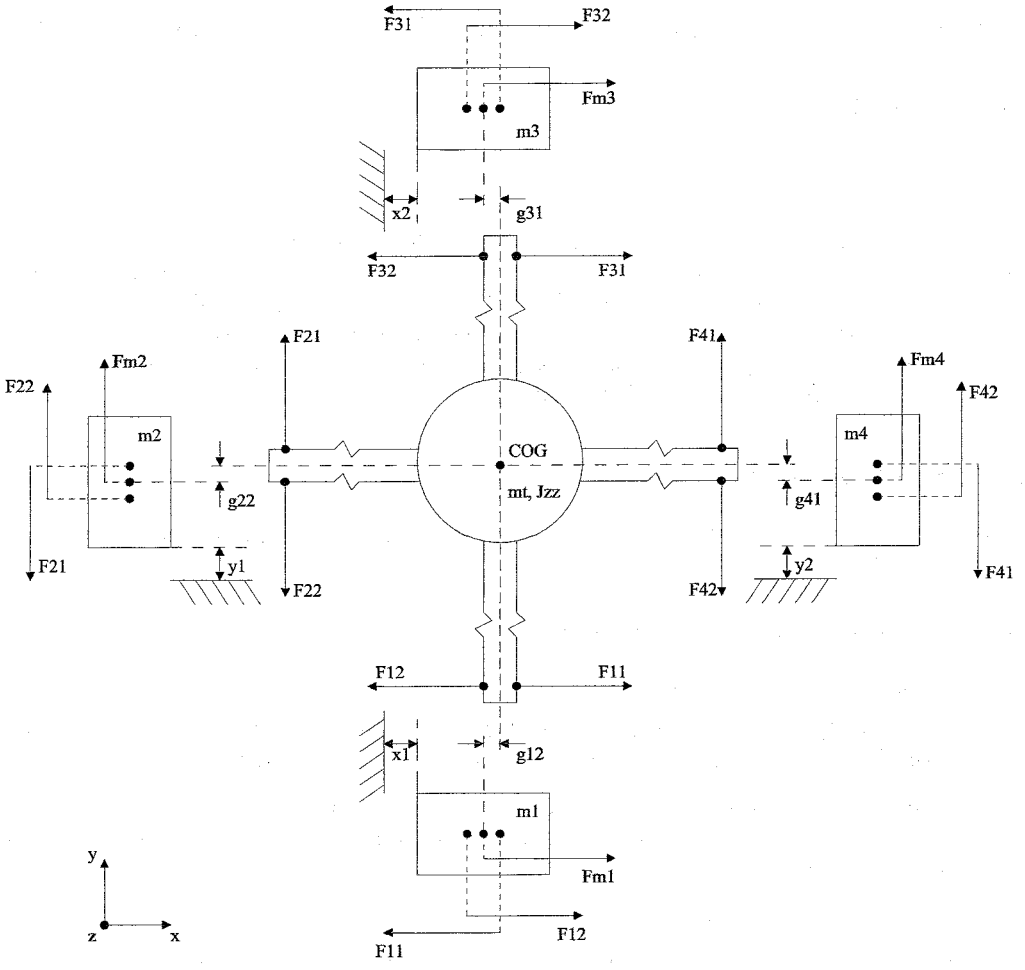


Figure 6.1: Free body diagram

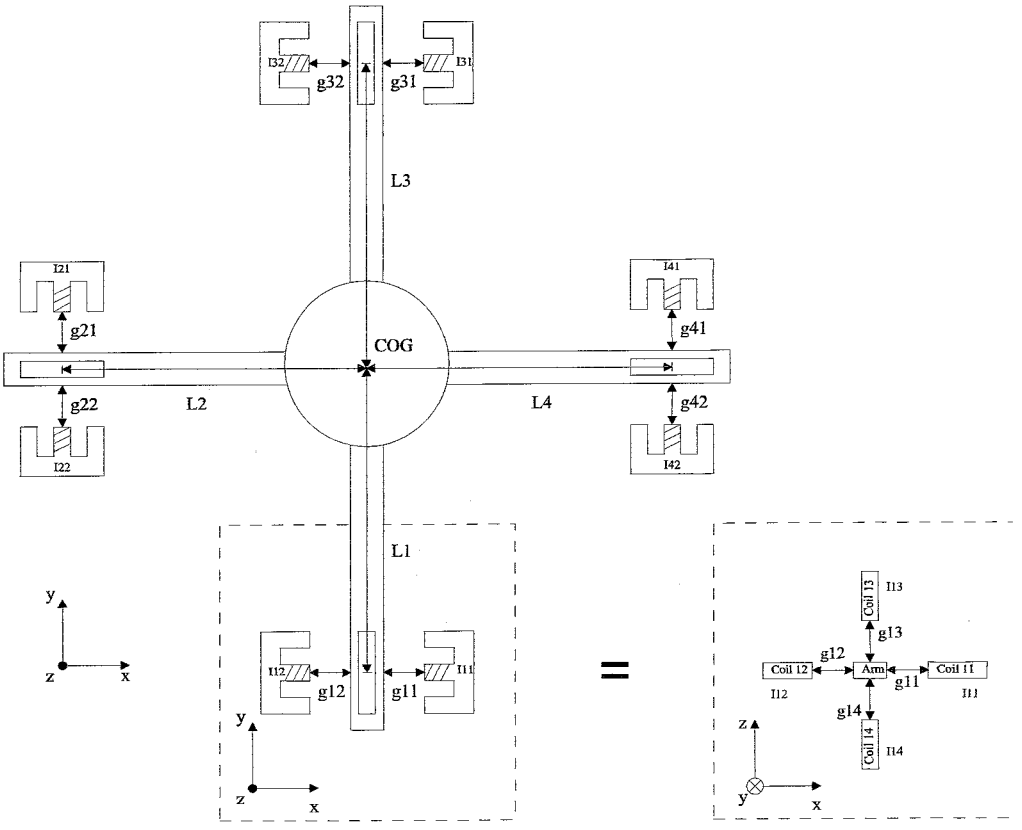


Figure 6.2: Geometry

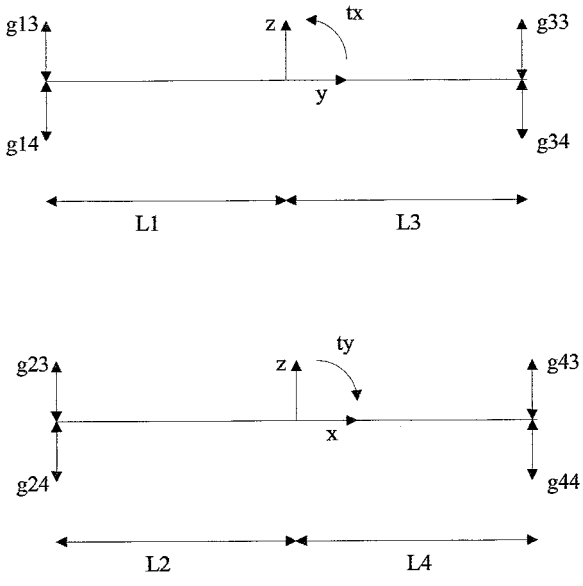


Figure 6.3: Gaps

••The physical system has 8 capacitive sensors, measuring the air gaps g_{11}, g_{21}, g_{32}, g_{42}, g_{14}, g_{24}, g_{34} and g_{44}. The complementary gaps are computed using the geometry of the system. For example: $g_{11} + g_{12} = 2g_{nx}$. Here, $g_{11} = g_{12} = g_{nx}$ is the nominal gap when the tray is in centered position (all states 0).

- The DOF's of the center of gravity (COG) are measured with respect to a reference frame that is fixed to ground. The origin of this frame coincides with the COG of the tray when it is in centered position.

- In x and y direction, distinction is made between long range motion (x_L, y_L) of the linear motors, and short range motion (x_S, y_S) of the tray with respect to the linear motors. The overall translations of the tray are $x_T = x_L + x_S$ and $y_T = y_L + y_S$.

••The linear motors move in pairs. It is assumed that travel of master and slave is equal ($x_1 = x_2 = x_L$ and $y_1 = y_2 = y_L$) so linear motors do not influence the angle θ_z.

••The origin of the long range motions x_L and y_L lies in the middle of their range.

- While computing moments acting on the tray, the influence of the DOF's $x_S, y_S, z, \theta_x, \theta_y$ and θ_z on the arm lengths is neglected. This is valid since the tray has only very small strokes (order 10^{-3} and 10^{-4}) in these DOF's. Only translations of the linear motors (strokes limited to plus or minus 0.104 [m]) have significant influence on the arm lengths. (In centered position, all magnetic actuators have an arm of 0.55 [m] w.r.t. the COG of the tray).

6.1.2 Equations of motion

The system model consists of 5 rigid body's: 4 linear motors and the tray. The linear motors have one translational DOF each. From Figure 6.1 it is easy to see that the equations of motion for these DOF's are:

$$\ddot{x}_1 = \frac{F_{m_1} + (F_{12} - F_{11})}{m_1} \tag{6.3}$$

$$\ddot{y}_1 = \frac{F_{m_2} + (F_{22} - F_{21})}{m_2} \tag{6.4}$$

$$\ddot{x}_2 = \frac{F_{m_3} + (F_{32} - F_{31})}{m_3} \tag{6.5}$$

$$\ddot{y}_2 = \frac{F_{m_4} + (F_{42} - F_{41})}{m_4} \tag{6.6}$$

The center of gravity of the tray has 6 DOF's with respect to ground:

$$\ddot{x}_T = \frac{(F_{11} - F_{12}) + (F_{31} - F_{32})}{m_T} \tag{6.7}$$

$$\ddot{y}_T = \frac{(F_{21} - F_{22}) + (F_{41} - F_{42})}{m_T} \tag{6.8}$$

$$\ddot{z} = \frac{(F_{13} - F_{14}) + (F_{23} - F_{24}) + (F_{33} - F_{34}) + (F_{43} - F_{44}) - m_T g}{m_T} \tag{6.9}$$

$$\ddot{\theta}_x = \frac{-L_1(F_{13} - F_{14}) + L_3(F_{33} - F_{34})}{J_{xx}} \tag{6.10}$$

$$\ddot{\theta}_y = \frac{L_2(F_{23} - F_{24}) - L_4(F_{43} - F_{44})}{J_{yy}} \tag{6.11}$$

$$\ddot{\theta}_z = \frac{L_1(F_{11} - F_{21}) - L_2(F_{21} - F_{22}) - L_3(F_{31} - F_{32}) + L_4(F_{41} - F_{42})}{J_{zz}} \tag{6.12}$$

Here $L_1 ... L_4$ represent the arm between each magnetic actuator and the COG. The arms can be approximated by:

$$L_1 = (L_0 + y_T)$$
$$L_2 = (L_0 + x_T)$$
$$L_3 = (L_0 - y_T)$$
$$L_4 = (L_0 - x_T)$$

L_0 is the arm length when the tray is centered ($L_1 = L_2 = L_3 = L_4 = L_0$).

6.1.3 Master-slave

The assumption that the translations of the slaves (motor 3 and motor 4) equal the translations of the masters (motor 1 and motor 2, respectively), imposes the following condition on the forces acting on the slaves:

$$F_{m_3} = m_3(\frac{F_{m_1} + (F_{12} - F_{11})}{m_1}) - (F_{32} - F_{31}) \tag{6.13}$$

$$F_{m_4} = m_4(\frac{F_{m_2} + (F_{22} - F_{21})}{m_2}) - (F_{42} - F_{41}) \tag{6.14}$$

6.1.4 Decoupling

In x- and y-direction, the total translation of the tray is the sum of the short-stroke translation x_S, y_S of the tray with respect to the linear motors, and the long-stroke translation x_L, y_L of the linear motors with respect to the ground:

$$x_T = x_L + x_S \tag{6.15}$$

$$y_T = y_L + y_S \tag{6.16}$$

Since the equations of motion for both the tray and the linear motors are known, the 2nd derivative of the short stroke follows:

$$\ddot{x}_T = \frac{F_{magn}}{m_T} \tag{6.17}$$

$$\ddot{x}_L = \frac{(F_{lin} - F_{magn})}{m_L} \tag{6.18}$$

$$\ddot{x}_S = \frac{F_{magn}}{m_T} - \frac{(F_{lin} - F_{magn})}{m_L} = \frac{(m_T + m_L)}{(m_T m_L)} F_{magn} - \frac{1}{m_L} F_{lin} \tag{6.19}$$

Translations x_S and y_S of the tray with respect to the linear motors are undesired. These translations reduce the air gaps, thus limiting the maximum stroke of rotation θ_z. The same translation of the tray can be achieved by translating the linear motors and keeping the crossmembers centered in the magnet boxes. So ideally $x_S = 0$.

This can be achieved as follows. Translations x_T and y_T are controlled using the magnetic actuators, which will lead to known reaction forces on the linear motors.

The equation of motion for the short stroke x_S is $m_L\ddot{x} = F_R$. The resultant force F_R required to maintain $x_S = 0$ can be computed using the linear controller designed in Chapter 3.

$$m_L\ddot{x}_S = F_R = \frac{(m_T + m_L)}{m_T} F_{magn} - F_{lin} \tag{6.20}$$

Now, the required force delivered by the linear motor F_{lin} can be determined easily.

$$F_{lin} = \frac{(m_T + m_L)}{m_T} F_{magn} - F_R \tag{6.21}$$

6.1.5 Measured variables vs. performance variables

Goal is to control the DOF's $x_T, y_T, z, \theta_x, \theta_y$ and θ_z of the center of gravity of the tray. At this moment, these DOF's cannot be measured directly. The necessary high-precision mirror and laser system will only be purchased once the machine design is permanent and budget allows. These DOF's have to be extracted from the available data using knowledge of the system geometry.

In the physical system, linear motor displacements x_1, x_2, y_1, y_2 are measured with a laser system, and the gaps $g_{11}, g_{21}, g_{32}, g_{42}, g_{14}, g_{24}, g_{34}$ and g_{44} are measured using capacitive sensors. Neglecting rotational cross terms, width and height of the arms, the measured gaps can be described as a function of the 6 DOF's of the tray (see also Figure 6.3).

$$
\begin{aligned}
g_{11} &= g_{nx} - (x_T - x_1) - L_1 \tan(\theta_z) \\
g_{21} &= g_{ny} - (y_T - y_1) + L_2 \tan(\theta_z) \\
g_{32} &= g_{nx} + (x_T - x_2) - L_3 \tan(\theta_z) \\
g_{42} &= g_{ny} + (y_T - y_2) + L_4 \tan(\theta_z)
\end{aligned}
$$

$$
\begin{aligned}
g_{14} &= g_{nz} + z - L_1 \tan(\theta_x) \\
g_{24} &= g_{nz} + z + L_2 \tan(\theta_y) \\
g_{34} &= g_{nz} + z + L_3 \tan(\theta_x) \\
g_{44} &= g_{nz} + z - L_4 \tan(\theta_y)
\end{aligned}
$$

So:

$$
\begin{bmatrix}
g_{11} - g_{nx} - x_1 \\
g_{21} - g_{ny} - y_1 \\
g_{32} - g_{nx} + x_2 \\
g_{42} - g_{ny} + y_2 \\
g_{14} - g_{nz} \\
g_{24} - g_{nz} \\
g_{34} - g_{nz} \\
g_{44} - g_{nz}
\end{bmatrix}
=
\underbrace{\begin{bmatrix}
-1 & 0 & 0 & 0 & 0 & -L_1 \\
0 & -1 & 0 & 0 & 0 & L_2 \\
1 & 0 & 0 & 0 & 0 & -L_3 \\
0 & 1 & 0 & 0 & 0 & L_4 \\
0 & 0 & 1 & -L_1 & 0 & 0 \\
0 & 0 & 1 & 0 & L_2 & 0 \\
0 & 0 & 1 & L_3 & 0 & 0 \\
0 & 0 & 1 & 0 & -L_4 & 0
\end{bmatrix}}_{S}
\begin{bmatrix}
x_T \\ y_T \\ z \\ \tan(\theta_x) \\ \tan(\theta_y) \\ \tan(\theta_z)
\end{bmatrix}
$$

There are 4 relations for the horizontal DOF's (x_T, y_T and $\tan(\theta_z)$) and 4 relations for the vertical DOF's (z, $\tan(\theta_x)$, and $\tan(\theta_y)$). They are dependent (the matrix has rank 6), so 6 independent relations can be established for 6 DOF's. Solving them gives $[x_T \quad y_T \quad z \quad \tan(\theta_x) \quad \tan(\theta_y) \quad \tan(\theta_z)]^T$, and subsequently θ_x, θ_y and θ_z can be derived.

6.2 Control of non-linear system

Due to delayed delivery of the current drive amplifiers, no thorough comparison between different control strategies could be made yet on the 1-DOF system. Based on temporary results, the following approach using sliding mode control of the feedback linearized system was chosen. The 6-DOF system can be described as follows:

$$x_1 = [x_T \quad y_T \quad z \quad \theta_x \quad \theta_y \quad \theta_z]^T$$

$$x_2 = \left[\dot{x_T} \quad \dot{y_T} \quad \dot{z} \quad \dot{\theta_x} \quad \dot{\theta_y} \quad \dot{\theta_z}\right]^T$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{21} \end{bmatrix} u$$

With:

$$a_{11} = [0]_{(6x6)} \qquad a_{12} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad a_{21} = [0]_{(6x6)} \qquad a_{22} = [0]_{(6x6)}$$

$$b_{11} = [0]_{(6x6)} \qquad b_{21} = \begin{bmatrix} \frac{1}{m_t} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{m_t} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{m_t} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{J_{xx_t}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{J_{yy_t}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{J_{zz_t}} \end{bmatrix}$$

With $u = [F_{x_{COG}} \quad F_{y_{COG}} \quad (F_{z_{COG}} - m_t g) \quad T_{x_{COG}} \quad T_{y_{COG}} \quad T_{z_{COG}}]^T$ the resultant forces and moments with respect to the center of gravity. Using forces and moments as input, the sliding mode algorithm is straightforward. Array S contains sliding surfaces for all 6 DOF's:

$$S = \dot{e} + 2\Lambda e + \Lambda^2 \int_0^t e \, dt$$

With $e = x_1 - x_{1_d}$ and

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_6 \end{bmatrix}$$

Define $A_{11} = (a_{21} + 2\Lambda a_{11} + \Lambda^2)$ and $A_{21} = (a_{22} + 2\Lambda a_{12})$:

$$\dot{S} = A_{11} e_1 + A_{21} e_2 + \hat{H}(u - u_d)$$

Substitution of a_{11}, a_{12}, a_{21} and a_{22} gives:

$$A_{11} = \Lambda^2$$

$$A_{21} = 2\Lambda$$

Now:

$$u = u_d - \hat{H}^{-1}(u_s + A_{11}e_1 + A_{21}e_2)$$

With u_d an array containing the desired inputs:

$$u_d = \hat{H}^{-1}(\dot{x}_{2d})$$

And u_s and array containing the switching terms:

$$u_s = -G_k \cdot \arctan(\Phi S)$$

With G_k an array containing the gain of the switching term for each DOF:

$$G_k = [g_{k_1} \quad g_{k_2} \quad g_{k_3} \quad g_{k_4} \quad g_{k_5} \quad g_{k_6}]^T$$

And Φ a diagonal matrix containing the boundary layer thickness for each DOF:

$$\Phi = \begin{bmatrix} \phi_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \phi_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \phi_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & \phi_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & \phi_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & \phi_6 \end{bmatrix}$$

6.2.1 Mapping

The computed control actions u represent forces and moments with respect to the COG. To translate these control actions to actuators inputs, a mapping will be used. Since a pair of opposed coils is needed to be able to produce attractive *and* repulsive forces, for this moment one pair of opposed coils is considered as one actuator. The first index is the number of the axis, the second index tells if the horizontally (h) or vertically (v) oriented pair of coils is considered. It is easy to see that the resultant forces and moments with respect to the tray's COG equal:

$$\begin{bmatrix} F_{x_{COG}} \\ F_{y_{COG}} \\ F_{z_{COG}} - m_T g \\ T_{x_{COG}} \\ T_{y_{COG}} \\ T_{z_{COG}} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & -L_1 & 0 & L_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & L_2 & 0 & -L_4 \\ L_1 & -L_2 & -L_3 & L_4 & 0 & 0 & 0 & 0 \end{bmatrix}}_{M} \begin{bmatrix} F_{1_h} \\ F_{2_h} \\ F_{3_h} \\ F_{4_h} \\ F_{1_v} \\ F_{2_v} \\ F_{3_v} \\ F_{4_v} \end{bmatrix} \tag{6.22}$$

This relation cannot be inverted directly. When the control actions $F_{x_{COG}} \ldots T_{z_{COG}}$ are computed, the approach proposed by [6] is used to determine the force each pair of coils has to deliver. Use:

$$\begin{bmatrix} F_{1_h} \\ \ldots \\ F_{4_v} \end{bmatrix} = M^* \begin{bmatrix} F_{x_{COG}} \\ \ldots \\ T_{z_{COG}} \end{bmatrix} \tag{6.23}$$

With:

$$M^* = \begin{bmatrix} \frac{M_1}{||M_1||_2^2} & \frac{M_2}{||M_2||_2^2} & \cdots & \frac{M_6}{||M_6||_2^2} \end{bmatrix}$$

With M_1 the first row of M, etc. , such that MM^* is a diagonal matrix with ones on the diagonal when the tray is centered.

So after the control actions with respect to the COG are computed, they are translated to a force for every pair of opposed coils. After this, the previously used nonlinear transformation is used to compute the input current to each coil. Since each coil can only exert attractive forces, on any moment only one coil of an opposite pair is active (depending on the sign of the required force). For example, for the horizontal coils in magnet box 1:

$$I_{11} = g_{11}\sqrt{\frac{F_{1_h}}{k_{11}}} \quad \text{and} \quad I_{12} = 0 \quad \text{for} \quad F_{1_h} > 0 \tag{6.24}$$

$$I_{12} = g_{12}\sqrt{\frac{F_{1_h}}{k_{12}}} \quad \text{and} \quad I_{11} = 0 \quad \text{for} \quad F_{1_h} < 0 \tag{6.25}$$

6.2.2 Simulations

A typical application of this system is high-precision positioning, in photo-lithography or the production of Micro Electro Mechanical Systems (MEMS). Some specifications with respect to maximum positioning errors are given in [10], but no information is available with respect to typical reference trajectories. In simulations, smooth (3rd order profile) steps were chosen as setpoint for each DOF. The amplitude of the steps is limited to half of the maximum stroke for each DOF, to make sure no mechanical contact occurs between the crossmembers and the magnet boxes.

Settings

- Since the amplifiers for the magnetic actuators have limited bandwidth, they are modelled as a first order low-pass filter with a pole at 100 Hz. This is a conservative estimate. The actuators saturate at 3.0 [A] input current.
- Setpoints used are 'smooth' steps, which means acceleration is a linear function of time. The tray has an initial z-displacement of $-6.35 \cdot 10^{-4}$ [m], since it initially rests at the lower coils. Step commands are given after 1.0 [s], so the tray can be lifted first and settle at $z = 0$ [m].
- Better results are achieved when the steps in x_T,y_T don't coincide with the step in θ_z. Reason is, that with θ_z = 0 all gaps have nominal width, which gives room for overshoot in x and y. If the cross is simultaneously rotated, half of the air gaps become smaller than the nominal distance gnx, leaving less room for overshoot. So it seems logic *first* to perform the translations in x and y, and to adjust the other DOF's after that.

Results

The sliding mode controllers for the 6 DOF's of the tray have been tuned such that the closed loop system has short settling time and low residual error. Gains were increased to improve performance until the system started to chatter, and than reduced a little.

The step time for x_T and y_T can not be taken too short. Stiffness of the sliding mode controllers is limited, so fast acceleration of the tray can leads to mechanical contact between tray an linear motors which is obviously undesired.

Simulations show that the mapping proposed in the previous section is suited to distribute the control actions over the 20 coils. When the tray is centered, MM^* is a diagonal matrix with 1's at the diagonal. However, if the tray moves away from the center, non-zero terms occur outside the diagonal of MM^*. This basically means that the control action for a certain DOF influences other DOF's as well. These cross-terms can be seen as disturbance.

Since no passive magnetic suspension is available, the vertical actuators have to provide a large force to carry the weight of the tray. This force is an order 100 higher than the other forces. When the tray moves off-center (motion in x_T and y_T, after 1 second), the cross-terms between the vertical force and the DOF's θ_x and θ_y increase from 0 to 0.3. The effect of this disturbance on θ_x and θ_y can clearly be seen in Figure 6.7 and Figure 6.8. This severely limits the maximum vertical speed in x_T and y_T.

If the tray moves faster in the horizontal plane, the disturbance on θ_x and θ_y is larger and leads to instability. Note that this problem is solved when a passive magnetic suspension carries the weight of the tray.

DOF	Start time [s]	Step duration [s]	Amplitude [m]
x_T	1.0	0.10	0.03
y_T	1.0	0.10	0.03
z	2.0	0.05	$3.0\ 10^{-4}$
θ_x	2.0	0.05	$3.0\ 10^{-4}$
θ_y	2.0	0.05	$3.0\ 10^{-4}$
θ_z	2.0	0.05	$4.0\ 10^{-4}$

Table 6.1: Setpoints

DOF	Labda	Φ	G_k
x_T	15	$1.0\ 10^{-1}$	10
y_T	15	$1.0\ 10^{-1}$	10
z	10	$1.0\ 10^{-2}$	15
θ_x	15	$7.0\ 10^{-3}$	10
θ_y	15	$7.0\ 10^{-3}$	10
θ_z	20	$1.0\ 10^{-2}$	15

Table 6.2: Controller gains

The linear controllers have the gains as used in Chapter 3.

6.3 Conclusion

A system model was made and a controller based on temporary results was designed. Simulated distribution of the control action for 6-DOF's to 20 coils works properly when linear motors are centered. This indicates that the approach using a mapping is suited. When the tray moves off-center, the quality of the distribution decreases and cross-terms are introduced. This implies that the control action for a certain DOF also affects other DOF's. In principle the robustness of the controller is high enough to deal with this disturbances. However, the vertical forces are large since they have to carry the weight of the tray. The disturbing effect of the vertical forces on the DOF's θ_x and θ_y is large, and limits the maximum speed of the tray in X- and Y-direction. This problem is solved when a magnetic suspension becomes available.

The force feed-forward leads to a good decoupling between the long-stroke and short-stroke motion, since x_S and y_S stay very small.

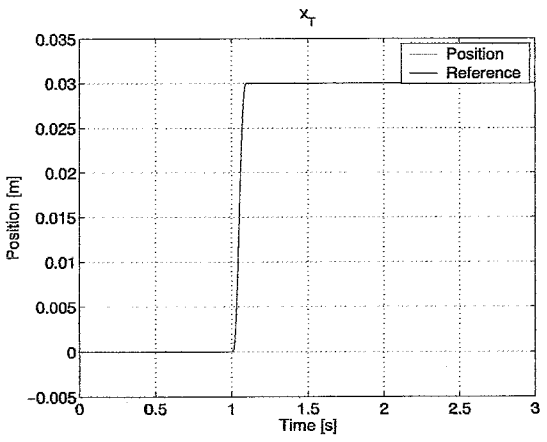


Figure 6.4: x_T: position and reference

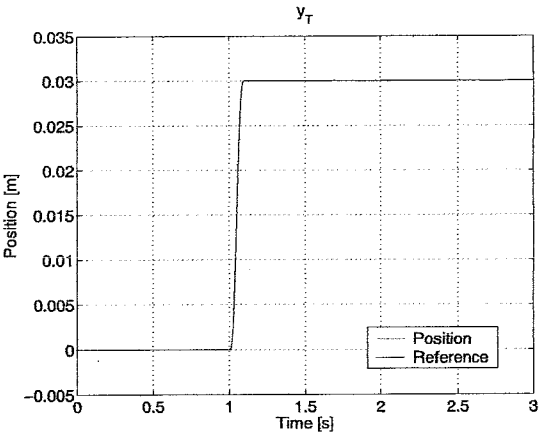


Figure 6.5: y_T: position and reference

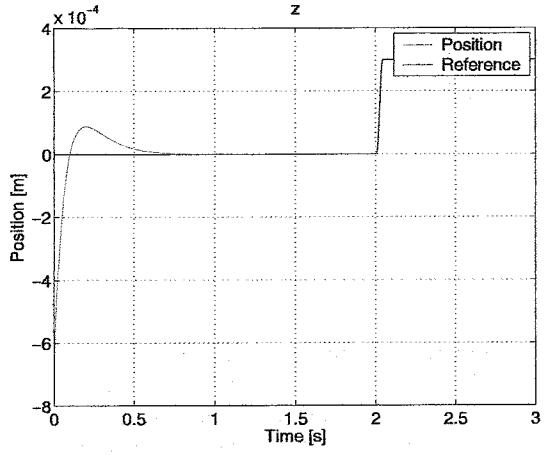


Figure 6.6: z: position and reference

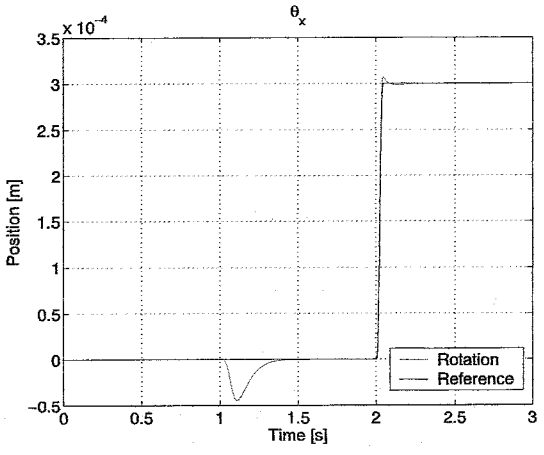


Figure 6.7: θ_x: rotation and reference

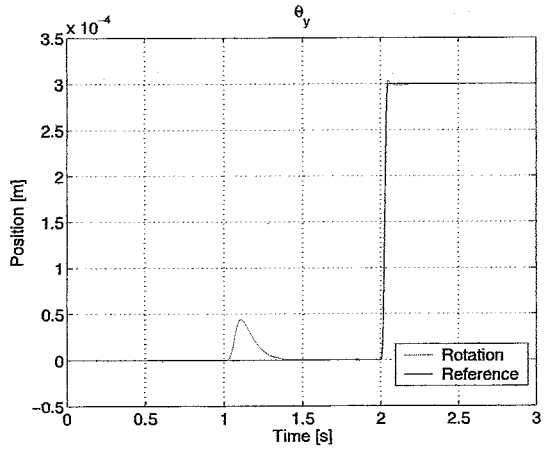


Figure 6.8: θ_y: rotation and reference

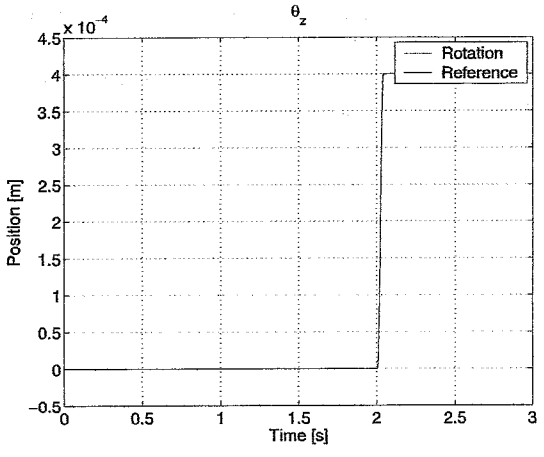


Figure 6.9: θ_z: rotation and reference

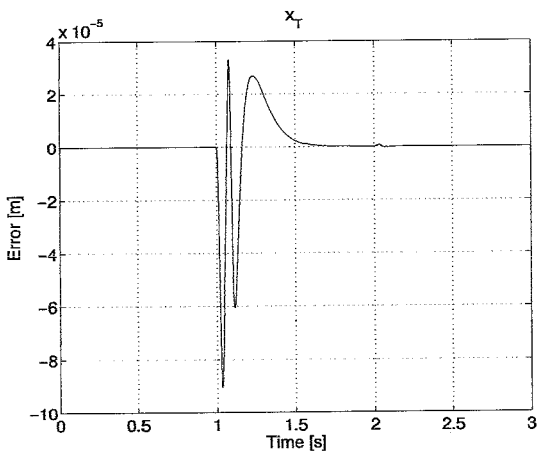


Figure 6.10: x_T: error

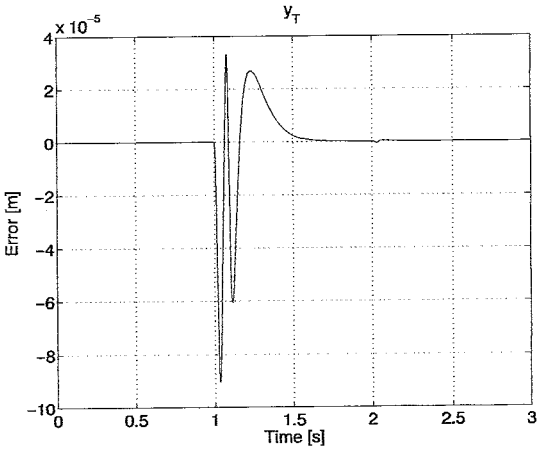


Figure 6.11: y_T: error

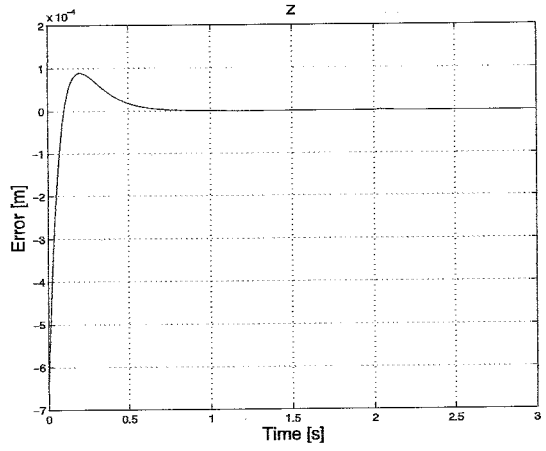


Figure 6.12: z: error

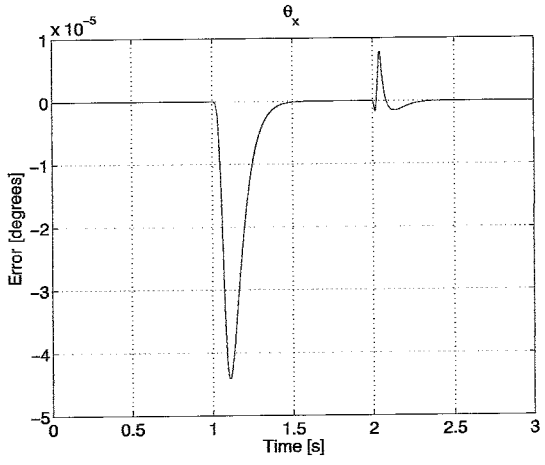


Figure 6.13: θ_x: error

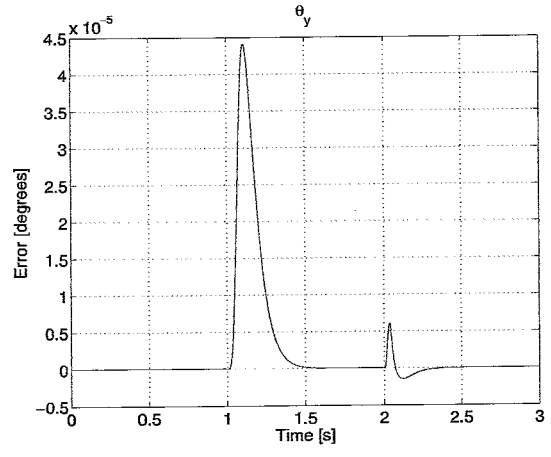


Figure 6.14: θ_y: error

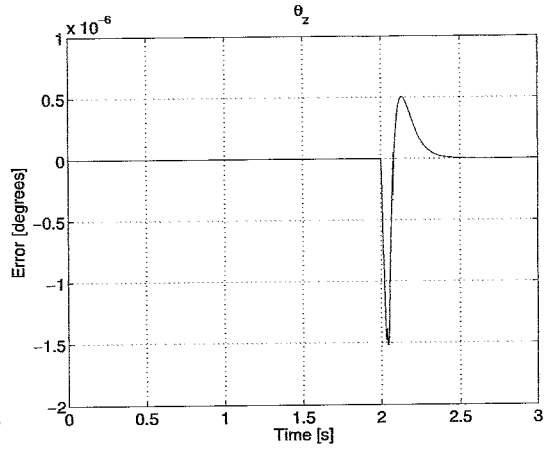


Figure 6.15: θ_z: error

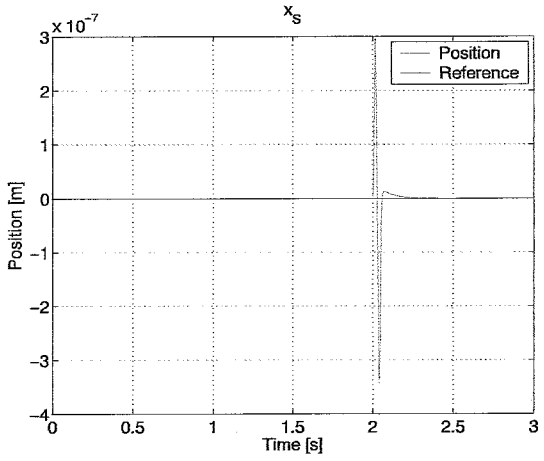


Figure 6.16: x_S: error

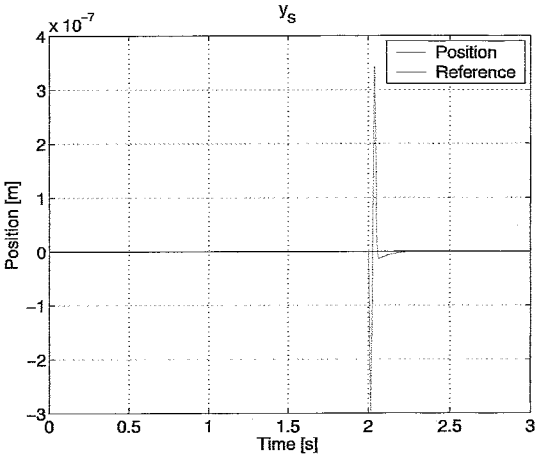


Figure 6.17: y_S: error

Chapter 7

Conclusion and recommendations

7.1 Conclusion

During this project, several steps towards the design and implementation of MIMO-control for the 6-DOF positioning stage were made. For control of the linear motors, hardware was assembled and electronics were connected. Preparing the laser system for position feedback involved replacement and (re)design of several components, addition of kinematic mounts and debugging. In addition to alignment of the laser and setting up the control software this took several months. Once the system was operational, system identification of the linear motors was straightforward. A linear controller was designed and implemented. The closed loop bandwidth is over 50 [Hz], while the steady state error is limited to 50 nanometer.

To gain understanding about the behavior and control of the magnetic actuators, a 1-DOF setup was made in which one crossmember is actuated by one magnetic actuator. A theoretical force model for the magnetic actuators was derived, and experiments in the 1-DOF setup showed that this model closely matches the empirical results.

Linear and nonlinear control strategies for control of the 1-DOF setup were simulated and implemented. This part of the project was delayed significantly, since the originally designed current drive amplifiers were not suited due to low bandwidth and failure. Design and manufacturing of new amplifiers took much time. Debugging and programming of a DSP for control implementation was time consuming too.

Due to limited availability of amplifiers, a thorough analysis of control of the 1-DOF setup, which was one of the main goals of this project, could not be finished. A linear controller was implemented with very poor results. Best results were obtained using a sliding mode controller in combination with feedback linearization of the system. The closed loop system is stable for a large operating range, only close to the upper actuator it is not. The system can track a low frequency reference signal, but improvement is definitely required. The steady state positioning error is well below 1 micron.

A start was made with the control of the 6-DOF system. A model was made using the results from system identification of the linear motors and the magnetic actuators. Long stroke and short stroke were decoupled, and the issue of controlling 6 DOF's with 20 actuators was addressed using a mapping to distribute the control signals. Some simulation results were shown, but more information considering setpoints and system specifications is required.

Some steps towards implementation have been made, such as ordering equipment and writing code to read parallel laser data. Many other issues will have to be addressed before a implementation is possible (see Section 7.2).

Considering the results so far, it seems very hard to meet the design specifications. Although the object of [10] was design of a ultra high position system as used in photolithography, the current system seems more suited as a high precision gantry system as used for flat panel inspection or electronic assembly.

7.2 Recommendations

A lot of work has to be done before the system performance even gets close to the design specifications. First of all, assembly of the hardware has to be completed. Design of the passive magnetic suspension has to be finished, so it can be manufactured. A second laser system is necessary to measure the position of the tray with high accuracy. This also requires a high precision mirror at the bottom of the tray. Other hardware necessary to get closer to the specified accuracy includes a faster DSP with better noise properties. Fluctuations of temperature and humidity in the lab should be prevented.

In order to finish the design and assembly of the hardware, some design principles should be investigated further. During the past year, new ideas considering the metrology frame were born. These ideas include a different setup for the laser system, and abandon the principle of metrology mapping. These principles were not within the scope of this project, but definitely require attention.

From a control point of view, a lot of interesting work has to be done. Control of the magnetic actuators is far from perfect. Due to delayed delivery of equipment this could not be investigated thoroughly yet. Now the hardware is all operational, this step can be performed.

Another important issue is the integration of the controllers for linear motors and magnetic actuators. At this moment, control for the linear motors runs on a CNC-card, while the magnetic actuators are controlled with a DSP. Real time communication between a host program and the CNC-card has been established, but communication between the host program and DSP is not possible yet. This is an absolute requirement for successful implementation of one control system that controls both types of actuators.

If these steps are performed, a control system for the 6-DOF system can be implemented. Only at that moment performance of the complete 6-DOF system can be measured in terms of accuracy and speed. From there, performance should be improved iteratively.

Bibliography

[1] AEROTECH Inc., "BAL linear amplifier series user's manual V1.5", Pittsburgh (PA) USA, May 23 2002

[2] AEROTECH Inc., "BB501 interface board manual, V1.5", Pittsburgh (PA) USA, May 2 2001

[3] AEROTECH Inc., "The UNIDEX 500 motion controller and windows software, operation & technical manual V1.3a", Pittsburgh (PA) USA, November 17 2000

[4] AEROTECH Inc., " U-channel linear motors user's manual V1.8", Pittsburgh (PA) USA, May 29 2002

[5] AEROTECH Inc., Personal communication (mr. Ratin and mr. Fry), December 2003 - May 2004

[6] Boeij, J. de "Electrodynamic magnetic levitation: NASA's vision for the future", Traineeship report, Eindhoven University of Technology (CST 2003.55), 2003

[7] Charara, A., De Miras, J. and Caron, B. "Nonlinear control of a magnetic levitation system without premagnetization", IEEE transactions on CST, vol 4 no. 5, September 1996

[8] DeCarlo, R.A., Zak, S.H. and Drakunov, S.V. "Variable structure, sliding mode controller design" (from 'The control handbook [page 941-951], edited by W.S. Levine, CRC Press, 1996)

[9] Dehmelt, Klaus. Personal communication, September 2003 - September 2004.

[10] Fevre, L.J.P. "Design and construction of a 6 DOF positioning system with nanometer accuracy", M.Sc. Thesis, Florida Institute of Technology, December 2003

[11] Gutierrez, H.M. and Fevre, L.J.P. "Design and construction of a 6-DOF positioning system with long range motion in XY and nanometer resolution usign magnetic servo-levitation", presented at MIT, April 2004

[12] Gutierrez, H.M. and Ro, P.I. "Magnetic servo levitation by sliding mode control of non-affine systems with algebraic input invertibility"

[13] Gutierrez, H.M. and Ro, P.I. "Sliding mode control of a nonlinear-input system: application to a magnetically levitated fast-tool servo", IEEE transactions on Industrial Electronics, vol 45 no. 6, December 1998

[14] Jager, A. de "Advanced control" (lecture sheets), TU/e 2001

[15] Levine, J., Lottin, J. and Ponsart, J. "A nonlinear approach to the control of magnetic bearings", IEEE transactions on CST, vol 5 no. 4, September 1996

[16] Lindlau, J.D. and Knospe, C.R. "Feedback linearization of an active magnetic bearing with voltage control", IEEE transactions on CST, vol 10 no. 1, January 2002

[17] Ludwick, Stephen J., Trumper, D.L., Holmes, M.L. "Modeling and control of a six DOF magnetic/fluidic motion control stage", IEEE transactions on CST, vol 4 no. 5, September 1996

[18] Sadiku, Matthew N.O. "Elements of Electromagnetics", 3rd edition, Oxford University Press, New York, 2001

[19] Scherpen, J.M.A., Kerk, B. van der, Klaassens, J.B., Lazeroms, M. and Kan, S.Y. "Nonlinear control for magnetic bearings in deployment test rigs: Simulation and experimental results", Proc. of the 37th IEEE conference on Decision & Control, Tampa, FL USA, December 1998

[20] Slotine, J. and Li, W. "Applied nonlinear control", Prentice Hall, London, 1991

[21] Trumper, D.L., Olson, S.M. and Subrahmanyan, P.K. "Linearizing control of magnetic suspension systems", IEEE transactions on CST, vol 5 no. 4, July 97

[22] Visser, E.E. "Modelling and linear control of a magnetically suspended arm", Traineeship report, Eindhoven University of Technology (CST 2003.120), 2003

[23] Woodson, H.H. and J.R.Melcher Jr., "Electromechanical Dynamics Part 1", New York, Wiley, 1968

[24] Zygo Corporation, "ZMI500 series Interferometer system with 501 measurement board", Middlefield (CT) USA, 2004

Appendix A

Instrumentation for linear motors

A.1 Pinouts

This appendix gives pinouts of custom-made connections.

Pin	Signal
1	+ 5 V
2	Ground
3	Hall A
4	Hall B
5	Hall C
6	nc
7	nc
8	nc
9	nc

Table A.1: Motor feedback

Pin	Wire	Signal
1	red	+ 5 V
2	black	Ground
3	green	CW limit
4	blue	CCW limit
5	white	Home
6	nc	nc
7	nc	nc
8	nc	nc
9	nc	nc

Table A.2: Switches

Pin	Wire	Signal
1	black	Phase A
2	red	Phase B
3	white	Phase C
4	green/yellow	Common

Table A.3: Motor power

A PCB was designed to connect the parallel output of both enclosures to the DSP's baseboard digital I/O and the DIG32 module's digital I/O. The data from enclosure 1 (laser axis 1 and 2) is read using the baseboard DIO, the data from enclosure 2 (laser axis 3 and 4) is read using the DIG32 module DIO. Both enclosure outputs are 44 pins with identical pin-out, the two digital I/O's are both 50 pins and only have different grounds. The PCB is used to route the signals from the enclosure output to the right pins of the DIO's. The pin-out of connectors on the PCB is as follows:

	Enclosure parallel output	DSP Digital I/O	DIG32 module
Signal	Pin	Pin	Pin
Data bit 0...11	1...12	1...12	1...12
A0	13	25	25
A3	14	28	28
IOE	15	31	31
Data bit 12...23	16...27	13...24	13...24
A1	28	26	26
A4	29	29	29
gnd	30...39	49	50
SCLK0	40	nc	nc
RESET	41	30	30
A2	42	27	27
gnd	43	49	50
SCLK1	44	32	32

Table A.4: Pin-outs w.r.t. parallel laser data

A.2 Mounting of laser system

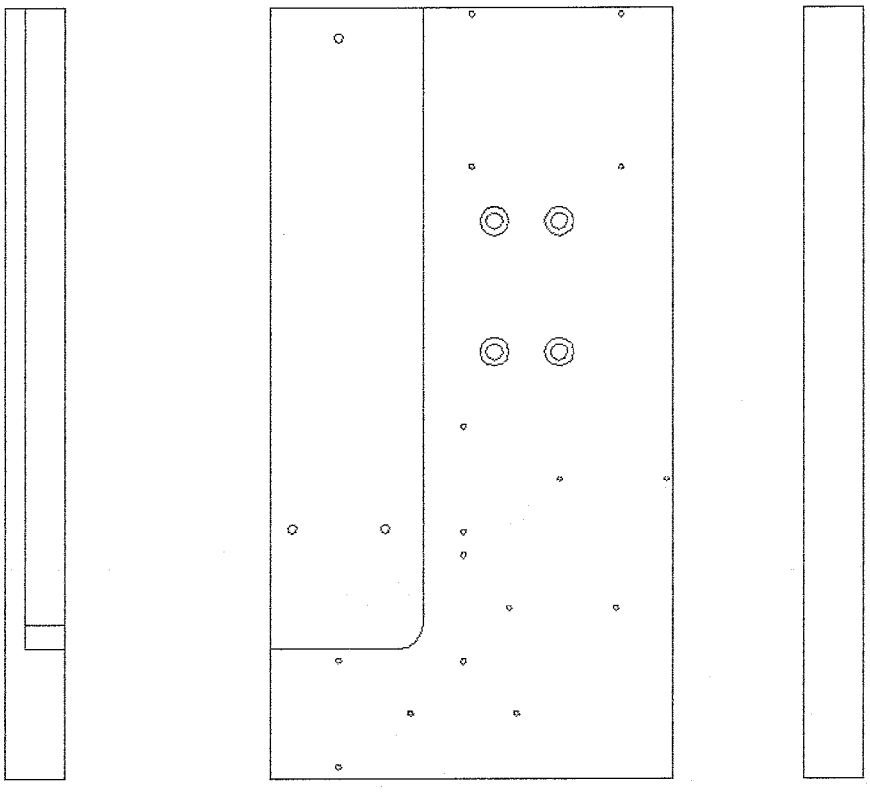


Figure A.1: Aluminum mounting block for optical components axis 1 & 2 and laser head

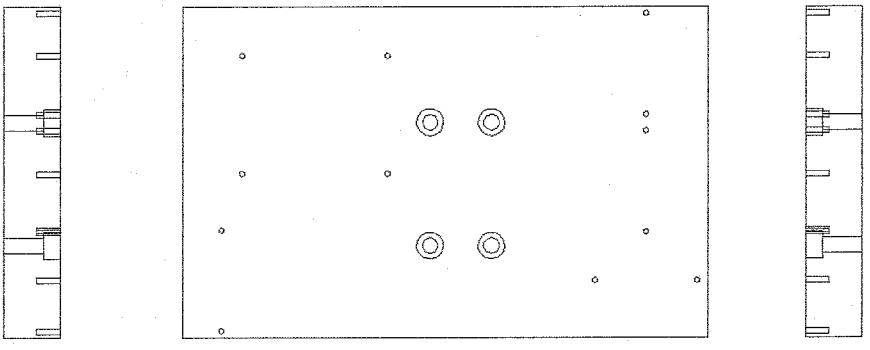


Figure A.2: Aluminum mounting block for optical components axis 3 & 4

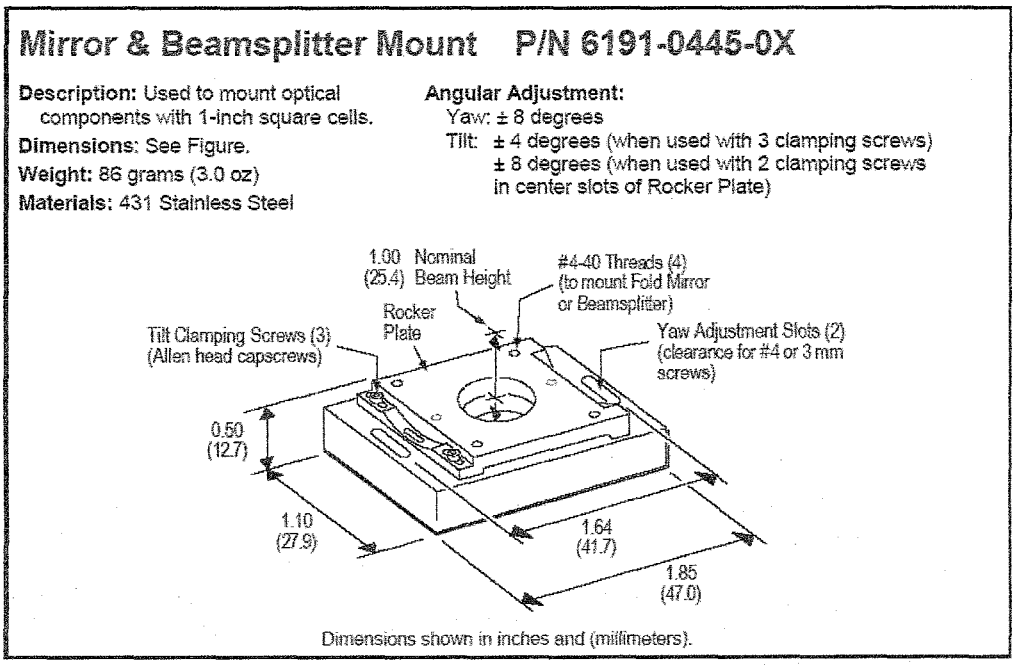


Figure A.3: Kinematic mount for mirror and beam splitter [24]

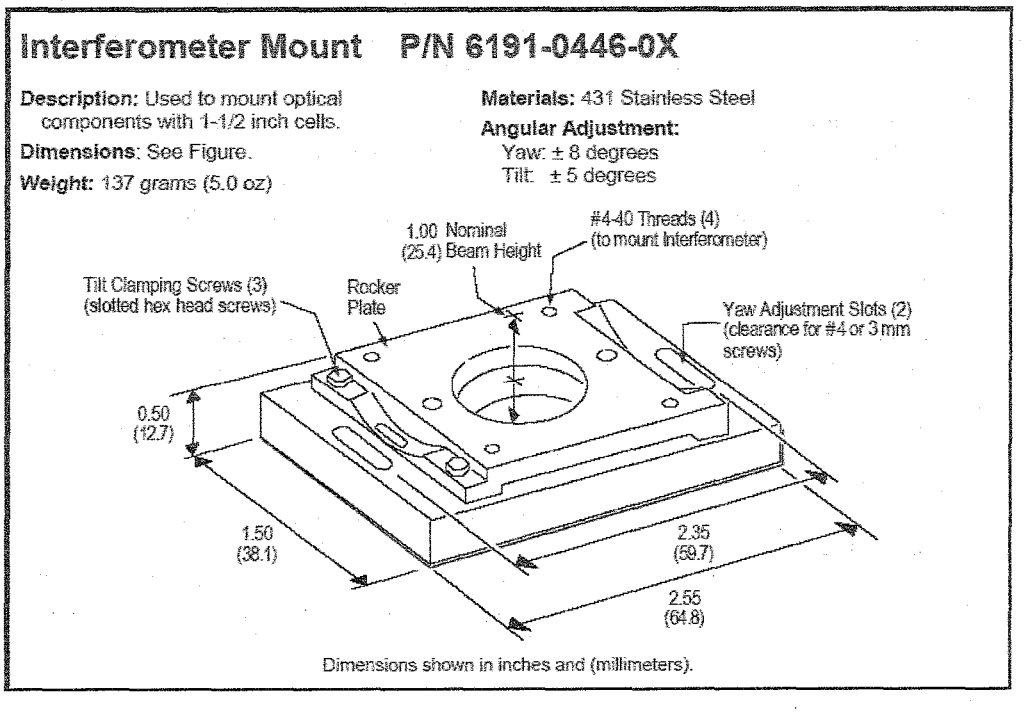


Figure A.4: Kinematic mount for interferometer [24]

A.3 Aligning laser system

This Appendix describes how to align all optical components of the laser system in a correct manner.

Alignment Tools: Penta prism, alignment card, alignment target with grid pattern.

1. Penta prism reflects light at 90 degrees irrespective of the relative angle between the prism and the input beam.
2. An alignment card has an adjustable hole for the input light to pass and can be used to see the position of the reflected beam from the interferometer without blocking the input beam.
3. The alignment target is a piece of paper or cardboard with fine grid pattern used to track the target beam drift on the target mirror.

A.3.1 Fold Mirrors and Beam Splitters

Mount all the optical components on their kinematic mounts so that each component's rocking plate rotates around the input beam. Mount the laser head and the first fold mirror. Adjust the screw on the kinematic mount towards the input laser beam so that the beam is centered in the input aperture of the fold mirror. Tighten the screw almost completely to avoid any further translations of the mount. Now put the penta-prism in the path of the beam to fold the beam. Adjust the position of the prism so that the light hits the alignment target on Axis 2 (see Figure A.5).

Now, move the axis across its entire range and note the horizontal drift in the input beam. Adjust the fold mirror until there is no horizontal drift in the input beam. Tighten the screw to secure the mount completely to the block. Now, the vertical drift has to be removed. Loosen the screws to adjust the pitch of the fold mirror. Tighten them back when there is no vertical drift in the input beam. Repeat the process till the input beam is stationary on the alignment target for the entire range of motion of the axis. Next, replace the prism with the second fold mirror and repeat the process to align the second mirror.

To align the first beam splitter, put the penta-prism on the right-angle output of the splitter (see Figure A.6). Repeat the above procedure by adjusting the pitch and yaw of the beam-splitter to stabilize both output beams on the target. Next, replace the prism with a fold mirror and repeat the procedure.

Continue the process to align all the mirrors and splitters on the aluminum block.

A.3.2 Interferometers

The next step is to align the interferometers. The input beam aperture on the interferometers is rather small after mounting the fiber optic pickup. Hence, it needs to be carefully aligned. Remove the fiber optic pickup from the interferometer and align its input side so that the input beam is on the opposite side to where the pickup gets mounted (see Figure A.7). A part of the beam may overflow the aperture. Tighten the screw on the input side to avoid translations.

Now, follow the previous instructions to align the yaw and pitch of the right-angled interferometer.

For a good pickup signal from the straight-through interferometer it is important to adjust pitch and yaw, even though the main output beam is unaffected by yaw. A very faint secondary spot is visible on the target plane, right of the main beam. The secondary beam is affected by yaw and pitch of the interferometer, and is used for alignment following the same procedure previously described. The secondary spot should not drift, for the entire range of the motion of the axis.

A.3.3 Target Mirror

Next step is alignment of the target mirror. Remove the alignment target from the target mirror. Adjust the optical table on which the laser head is mounted so that the input beam hits the target mirror on the left side, at center height. Adjust the mirror tilt and rotation so that output beam goes back into the aperture of the interferometer.

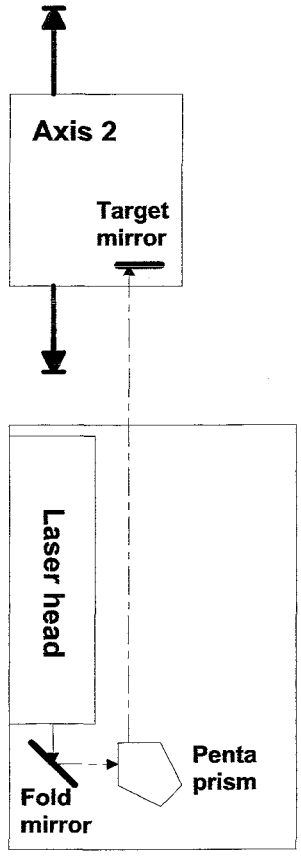


Figure A.5: Alignment of fold mirror

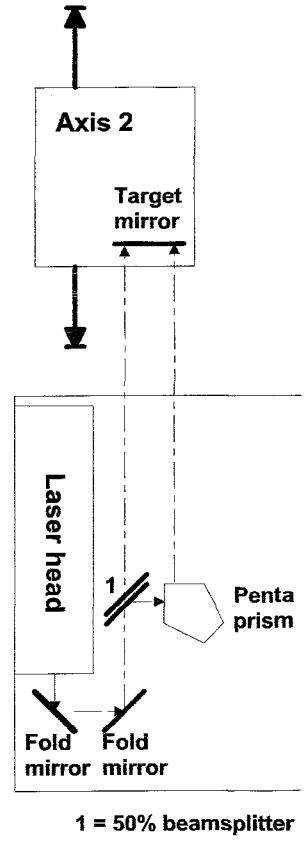


Figure A.6: Alignment of beam splitter

Now, place the alignment card before the interferometer and adjust the position so that none of the input beam going to the interferometer is blocked. An almost circular reflection should be visible on the alignment card to the right of the input hole when looking from target mirror. A second reflection should also be present on the alignment card which can be moved by changing the pitch and yaw of the target mirror. The two reflected beams should overlap each other completely for a good signal (see Figure A.9). Move the axis from end to end and observe the reflections, and adjust the target mirror so that reflections do not drift. Two spots should be observed on the target mirror, horizontally separated by about .25 [inch]. They should not move with respect to each other or w.r.t. the mirror for the entire trajectory of the axis (see Figure A.8).

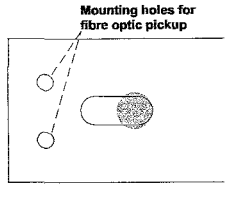


Figure A.7: Position of beam on input beam aperture

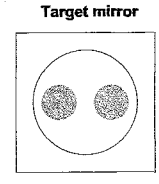


Figure A.8: Reflections on target mirror

A.3.4 Fiber Optic Pickup

The final step is to align the fiber optic pickup to the interferometer. Connect the pickup to the fiber optic cable and interferometer loosely to allow some adjustments. Check the output of the fiber optic cable on the other end by directing it to a piece of white paper as mentioned in [24] (page 7-11). The beam should be a solid spot and not a ring or a donut.

Connect the optic cable to the ZMI 501 enclosure and measure the voltage on the test point for the channel. Adjust the pickup to get a voltage in excess of 2.0 V for the entire range of the axis. *Note that the 2.4 V mentioned in the manual is not correct.* As long as the error light on the enclosure is off, the signal is fine. The voltage should be almost constant for the entire range of the motion. If the voltage varies more than 100mV, this indicates misalignment.

Once a constant voltage is achieved for the entire range, screw in the pickup completely. This step is difficult and requires patience. Sometimes, using 1 screw instead of 2 is better to maintain satisfactory output voltage.

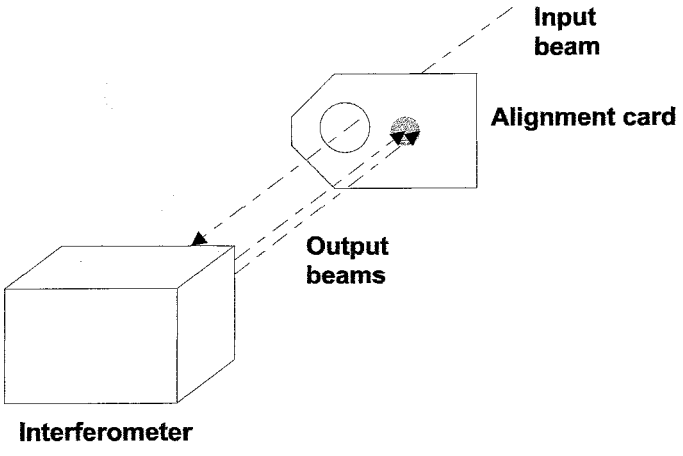


Figure A.9: Overlapping beams for whole axis

A.3.5 Feedback and dip switch settings

Finally, check the feedback in the U500 software. The axis should move for about 208 mm and should be constant for every move. The right-angled interferometer gives opposite output as compared to the straight through. Hence, the right-angled interferometer gives increasing feedback when the target mirror is moving closer to it. As explained in [24] (page 3-3) switch S8 can be used to swap the direction of the quadrature/serial output going to the control software. The motor expects an increasing feedback in the 'clockwise' direction of the motion (the side where the wires of the motor come out). Therefore, adjust switch S8 for all the axes accordingly. An incorrect feedback will make the axis go unstable by just enabling it. Also note that the position of switch S8 does not affect the parallel output.

Appendix B

Actuator dimensions and equivalent circuit

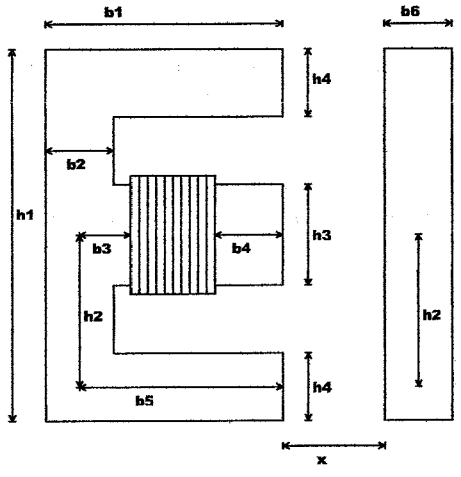


Figure B.1: Actuator dimensions

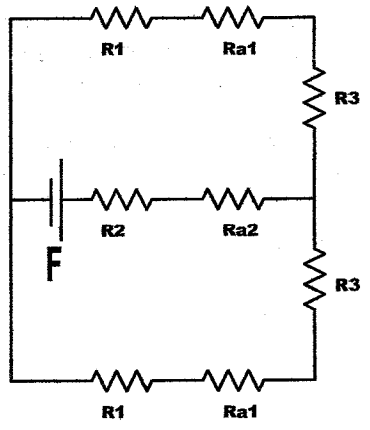


Figure B.2: Equivalent circuit

	Length [inch]	Length [mm]
b_1	1.753	44.5
b_2	0.455	11.6
b_3	0.272	6.9
b_4	0.325	8.3
b_5	1.526	38.8
b_6	0.500	12.7
h_1	2.628	66.8
h_2	1.098	27.9
h_3	0.631	16.0
h_4	0.433	11.0
d_1	0.648	16.5

Table B.1: Dimensions of magnetic actuator

Four different surfaces can be distinct through which the flux paths pass:

$$\begin{aligned} S_1 &= d_1 \cdot h_3 \\ S_2 &= d_1 \cdot h_4 \\ S_3 &= d_1 \cdot b_2 \\ S_4 &= d_1 \cdot b_6 \end{aligned}$$

For each part of the actuator, the reluctance can be computed using Equation 4.9:

$$\begin{aligned} R_1 &= \frac{1}{\mu_0 \mu_r}\left(\frac{b_3}{S_1} + \frac{h_2}{S_3} + \frac{b_5}{S_2}\right) \\ R_2 &= \frac{1}{\mu_0 \mu_r}\frac{b_4}{S_1} \\ R_3 &\approx \frac{1}{\mu_0 \mu_r}\frac{h_2}{S_4} \\ R_{a_1} &= \frac{1}{\mu_0}\frac{x}{S2} \\ R_{a_2} &= \frac{1}{\mu_0}\frac{x}{S1} \end{aligned}$$

The reluctances R_1, R_2 and R_3 are constant, while R_{a_1} and R_{a_2} are dependent on x, the magnitude of the air gap. Since eventually only very small rotations of the 'I' with respect to the 'E' are considered, it is assumed that the air gaps at all of the 3 legs of the 'E' are equal.

Using the well-known laws for electrical circuits, an equivalent reluctance R_T for the total circuit can be derived. A distinction is made between the constant reluctance R_C and the term depending on air gap, R_X:

$$\begin{aligned} R_T &= R_C + R_X \\ R_C &= \tfrac{1}{2}R_1 + \tfrac{1}{2}R_2 + R_3 \\ R_X &= \tfrac{1}{2}R_{a1} + R_{a2} \end{aligned}$$

Appendix C

Code

C.1 C-code for reading parallel laser data

```
#include "periph.h"
#include "stdio.h"
#include <math.h>

#define A0 24
#define A1 25
#define A2 26
#define A3 27
#define A4 28
#define RESET 0x1D
#define IOE 0x1E
#define SCLK1 0x1F

// Globals
volatile int module = 0;
unsigned int errors;
volatile unsigned int readback;

// Prototypes
void            test_dio();
void            test_base_dio();
void            read_laser();
void            pause();

//-----------------------------------
//     main() - Menu of test choices
//-----------------------------------
main()
{
    enable_monitor();
    enable_clock();
    enable_interrupts();

start:
    clrscr();
    gotoxy(35, 0);
    bold();
    printf("\7Laser Data test\n");
    normal();
    printf("\n\nCurrent module: %d", module);

    printf("\n\n  Main Menu\n");
    printf("1) Read Laser Data\n");
    printf("2) BaseBoard Digital I/O\n");
    printf("3) Module Digital I/O\n");
    printf("4) Select Module to Test\n");
    printf("5) Quit\n");
    printf("\nSelect test to run: ");

    switch (getchar())
    {
        case '1':
            read_laser();
            break;
        case '2':
```

```
            test_base_dio();
            break;
        case '3':
            test_dio();
            break;
        case '4':
            printf("\nModule Site [0-2]: ");
            scanf("%d", &module );
            printf("\n");
            break;
        default:
            printf("\n\nTesting Complete.");
            monitor();
            break;
    }
    printf("\n\nPress a key to return to main menu...");
    getchar();

    goto start;
}

//-------------------------------------------------
//     read_laser() - Read 32 bit data from laser
//-------------------------------------------------

void read_laser() {
    volatile unsigned int rtn;
    int col, row;
    volatile int counter=0, old_pos, pos, position, diff, first_time=1;

    //  set software triggering
    set_dig_trigger(DIO_TRIGGER_SOFT);

    clrscr();
    printf("Reading Laser telemetry data");
    cursor(OFF);

    clrscr();
    printf("\n\nPress any key to terminate test.");
    wherexy(&col, &row);

    // initialize all baseboard pins for output (control signal for laser)
    dig_dir(DIO_DIR_OUTPUT, DIO_DIR_INPUT, DIO_DIR_INPUT, DIO_DIR_INPUT);

    write_dig(0);
    write_dig_bit(RESET, 1);    //active low
    write_dig_bit(IOE, 0);      //active low

    //  Loop doing writes to Digital I/O

    while (!kbd_hit())
        {
            write_dig_bit(A0, 1);
            write_dig_bit(A1, 0);
            write_dig_bit(A2, 0);
            write_dig_bit(A3, 1);
            write_dig_bit(A4, 0);
            ns(20);
            write_dig_bit(SCLK1, 1);
            //ns(20);
            //write_dig_bit(IOE, 0);
            ns(120);
            rtn = read_dig();       //read data from laser
            pos=rtn%0x1000000;

            if (first_time) {
                diff=0;
                first_time=0;
            }
            else
               diff=pos-old_pos;

            old_pos=pos;

            if (abs(diff)>(pow(2,23))) {
            // overflow
                if (diff>0)
                    counter--;
                else
                    counter++;
```

```
            }

            position=counter*pow(2,24)+pos;
            position=position*1.235977184;
            position=position/1e6;

            write_dig_bit(SCLK1, 0);
            //write_dig_bit(IOE, 1);

            gotoxy(0, row+1);
            printf("Current value %10x Current displacement %d mm ", pos, position);

            /*
            ms(1);
            write_dig_bit(A0, 0);
            write_dig_bit(A1, 1);
            write_dig_bit(A2, 0);
            write_dig_bit(A3, 1);
            write_dig_bit(A4, 0);

            ns(20);
            write_dig_bit(SCLK1, 1);
            ns(20);
            write_dig_bit(IOE, 0);
            ns(120);
            rtn = read_dig();       //read data from laser
            velocity=rtn*1.235977184;;
            velocity=velocity/1000;

            write_dig_bit(SCLK1, 0);
            write_dig_bit(IOE, 1);

            gotoxy(0, row+2);
            printf("Current value %10x Current velocity %d um/s", rtn, velocity);
            */
        }
}

//-----------------------------------------------------------------------
//  test_base_dio()  --  Test Baseboard Digital I/O
//-----------------------------------------------------------------------

void test_base_dio(void)
{
    int          attr;
    int          i;
//    BaseboardDio* dio = (BaseboardDio*)&Periph->Dio;
    volatile unsigned int shadow = 0;
    volatile unsigned int rtn;
    volatile unsigned int err = 0;
    volatile unsigned int reps = 0;
    int col, row;

    clrscr();
    printf("\nTesting M6x baseboard digital...\n\n");
    printf("\n\nInstall Loopback Cable on Digital Connector...");
    pause();

    cursor(OFF);
    attr = get_attribute();
    //
    //  set software triggering
    set_dig_trigger(DIO_TRIGGER_SOFT);

    for (i=0; i<2; i++)
        {
        int highState, lowState;

        err = 0; reps = 0;
        clrscr();
        printf("\n\nPress any key to terminate test.");
        //
        // initialize Half for input, half output
        if (i==0)
            {
            highState=DIO_DIR_OUTPUT;
            lowState=DIO_DIR_INPUT;
            printf("\n\nLoopback High word to Low word: ");
            wherexy(&col, &row);
            }
```

```
        else
            {
            highState=DIO_DIR_INPUT;
            lowState=DIO_DIR_OUTPUT;
            printf("\n\nLoopback Low word to High word: ");
            wherexy(&col, &row);
            }
        dig_dir(highState, highState, lowState, lowState);
        //
        //  Loop doing writes to Digital I/O
        while (!kbd_hit())
            {
            int output;
            // Complement shadow register
            shadow ^= 0xffffffff;
            output = (shadow)?0xA5A5A5A5:0x5A5A5A5A;
            write_dig(output);

            if (shadow)
                set_attribute(RGB(255,0,0));
            else
                set_attribute(RGB(0,0,255));

            rtn = read_dig();

            gotoxy(col, row+1);
            printf("%10x  %10x", rtn, output);

            if (rtn != output)
                err++;
            reps++;
            if ((reps%100)==0)
                {
                gotoxy(col, row);
                printf("%d Errors in %d tries.\n", err, reps);
                }
            }
        getchar();  // eat the character
        gotoxy(col, row);
        set_attribute(attr);
        printf("%d Errors in %d tries.", err, reps);

        gotoxy(0, 20);
        pause();
        }

    set_attribute(attr);
    cursor(ON);

    clrscr();
    printf("\n\nRemove Loopback Cable from Digital Connector...");
    pause();

}

//-----------------------------------------------
//     test_dio() - Test all digital I/O bits
//-----------------------------------------------

void test_dio()
{
    int          attr;
    int          i;
    volatile unsigned int shadow = 0;
    volatile unsigned int rtn;
    volatile unsigned int err = 0;
    volatile unsigned int reps = 0;
    int col, row;

    clrscr();
    printf("Testing OMNIBUS DIG digital I/O...");

    //-------------------------------------------------
    //  Cable Loopback Tests
    //

    printf("\n\nInstall Loopback Cable on Digital Connector...");
    getchar();

    cursor(OFF);
```

```
    attr = get_attribute();

    for (i=0; i<2; i++)
        {
        int highState, lowState;

        err = 0; reps = 0;
        clrscr();
        printf("\n\nPress any key to terminate test.");
        //
        // initialize Half for input, half output
        if (i==0)
            {
            highState=DIO_DIR_OUTPUT;
            lowState=DIO_DIR_INPUT;
            printf("\n\nLoopback High word to Low word: ");
            wherexy(&col, &row);
            }
        else
            {
            highState=DIO_DIR_INPUT;
            lowState=DIO_DIR_OUTPUT;
            printf("\n\nLoopback Low word to High word: ");
            wherexy(&col, &row);
            }
        DIO_dig_dir(module, highState, highState, lowState, lowState);
        //
        //  Loop doing writes to Digital I/O
        while (!kbd_hit())
            {
            int output;
            shadow ^= (unsigned)-1;                    // Complement shadow register
            output = (shadow)?0xA5A5A5A5:0x5A5A5A5A;
            DIO_write_dig(module, output);

            if (shadow)
                set_attribute(RGB(255,0,0));
            else
                set_attribute(RGB(0,0,255));

            rtn = DIO_read_dig(module);

            gotoxy(col, row+1);
            printf("%10x  %10x", rtn, output);

            if (rtn != output)
                err++;
            reps++;
            if ((reps%100)==0)
                {
                gotoxy(col, row);
                printf("%d Errors in %d tries.\n", err, reps);
                }
            }
        getchar();  // eat the character
        gotoxy(col, row);
        set_attribute(attr);
        printf("%d Errors in %d tries.", err, reps);

        gotoxy(0, 20);
        getchar();
        }

    set_attribute(attr);
    cursor(ON);

    clrscr();
    printf("\n\nRemove Loopback Cable from Digital Connector...");
    getchar();
}

//-----------------------------------------------------
//  pause() -- wait for a key to be hit
//-----------------------------------------------------

void pause( void )
{
     printf("\n\nPress any key to continue...");
     getchar();
}
```

C.2 C-code for control of 1-DOF system

```
//-------------------------------------------------------------------------
// Feedback linearization in combination with SMC
// E.E.Visser
//-------------------------------------------------------------------------

#include "periph.h"
#include "stdio.h"
#include "dsp.h"
#include "math.h"

// Servo parameters:
#define CLK_FACTOR      24              // This number should not be changed
#define CLOCK_RATE      100000.0        // A/D conversion rate
#define DECIMATION      39              // Number of samples thrown out for each one kept
#define DAC_DELAY       100000          // At least 10800ns + max calc time
#define IN_SCALE        (10.0/32767.0)  // Input scaling (-10 to +10 V = 2^16)
#define OUT_SCALE       3276.7          // Idem for output
#define GAIN_SCALE      3.3333          // Amplifier gain (10 V input : 3 A output)
#define ts              4.0e-4          // Sample time

#define PAIR_MASK_OUT   0xff            // 16 output channels, 8 pairs (0xff)
#define PAIR_MASK_IN    0xf             //  8 input  channels, 4 pairs (0xf)

// System parameters:
#define Jxx             0.15            // [kg*m^2]
#define yy              0.3353          // [m]
#define ys              0.275           // [m]
#define gn1             5.81e-4         // [m], = 9.50*50e-6*yy/ys
#define gn2             5.50e-4         // [m], = 9.00*50e-6*yy/ys
#define m               1.694           // [kg]
#define g               9.81            // [m/s^2]
#define l               0.2456          // [m]
#define H               6.67            // = 1/Jxx;
#define pi              3.14159265358979

// Prototypes
#pragma CODE_SECTION(analog_isr, ".onchip");
#pragma INTERRUPT(analog_isr);
void analog_isr();
#pragma CODE_SECTION(DoServoCalc, ".onchip");
void DoServoCalc();
#pragma CODE_SECTION(Initialize_DSP, ".offchip");
void Initialize_DSP();
#pragma CODE_SECTION(Reference, ".onchip");
void Reference();
#pragma CODE_SECTION(active_pairs_out, ".offchip");
int active_pairs_out();
#pragma CODE_SECTION(active_pairs_in, ".offchip");
int active_pairs_in();

//-------------------------------------------------------------------------
// Global Data
//-------------------------------------------------------------------------

// General
near volatile AIO_PAIR Input[8];
near volatile AIO_PAIR Output[8];
near volatile AIO_PAIR ZeroOut[8];
unsigned int xrpt;
unsigned int xrpt_source;
near int ActivePairsOut;
near int ActivePairsIn;
near int module                 = 1;        // Servo16 module on site 1 of M67 baseboard
near int stopcondition          = 0;        // Emergency stop
near double freq                = 1/ts;     // Sample freq. controller

// Parameters for SMC:
near double labda               = 70.0;
near double phi                 = 15e-3;
near double gk                  = 4.0;

near double labda1              = 125.0;
near double labda2              = 80.0;
near double phi1                = 50.0e-3;
near double gk1                 = 7.0;

near double Ss                  = 0;        // Sliding surface
near double Us                  = 0;        // Switching term
```

```
near double Tx                  = 0;        // Torque around x-axis

// Error and reference
near double x,eold,err,edot,eint= 0;
near double r,rv,rddot,rvddot   = 0;

// Other
near double g1                  = 2*gn1;    // Initial upper gap [m]
near double g2                  = 0;        // initial lower gap [m]
near double V_INIT              = -9.0;     // Initial sensor voltage [V]
near double S2;                             // Sensor 2 [V] (vertical displacement)

near double I1,I2               = 0;        // Currents [A]
near double t                   = 0;        // Time [s]
near volatile int ClockCount    = 0;

near double K1,K2   = 4.55e-6;

near double P1      = -3.55671335314818;
near double P2      =  0.00855707663713;
near double P3      =  7.363349702380939e-007;

// near int odd=0;                          // For fine tuning of servo timing

//-------------------------------------------------------------------------
// main()
//-------------------------------------------------------------------------

main()
    {

    EnableCache();
    Initialize_DSP();

    // Enable interrupt on Servo16, allow data acquisition to start
    SERVO16_start(module);

    // Infinite loop, abort when user hits any key
while(!kbd_hit())
     {}
        stopcondition=1;
        printf("\n\nSTOP");
    }

//-------------------------------------------------------------------------
//  Initialize_DSP()
//-------------------------------------------------------------------------

void Initialize_DSP()
{
   int i;
    SERVO16_ID id_struct;

    ActivePairsIn   = active_pairs_in();
    ActivePairsOut  = active_pairs_out();
    EnableInterrupts();
    enable_monitor();

    gotoxy(30, 0);
    bold();
    printf("\7Servo 16 application\n");
    normal();

    // Get module site
    printf("\n\nServo 16 on module %d",module);
    cursor(OFF);

    // use A/D FIFO level interrupt
    xrpt = module ? EINT2_INTERRUPT : EINT0_INTERRUPT;
    xrpt_source = module ? EINT2_INT_SRC : EINT0_INT_SRC;

    // disable interrupt on module, disable FIFO data acquisition
    SERVO16_stop(module);

    //---------------------------------------
    //  read coefficients from IDROM and apply
    //---------------------------------------
    SERVO16_read_idrom(module, &id_struct);

    printf("\n\nIDROM coefficients used for gain and offset correction");
```

```
    // init gain and scale factor values to IDROM contents, all channels
        for(i = 0; i < 16; i++)
            {
            SERVO16_set_adc_gain(module, i, SERVO16_from_fixed(id_struct.adc_gain_coeff[i]));
            SERVO16_set_adc_offset(module, i, id_struct.adc_offset_coeff[i]);
            SERVO16_set_dac_gain(module, i, SERVO16_from_fixed(id_struct.dac_gain_coeff[i]));
            SERVO16_set_dac_offset(module, i, id_struct.dac_offset_coeff[i]);
            }

    //-----------
    //  Start DDS
    //-----------

    timebase(DDS_TIMER, CLOCK_RATE * CLK_FACTOR, DDS_CLOCK);

    //  Allow DDS to settle at new rate
    ns(1e7);


    //---------------------------
    //  Initialize Servo16 module
    //---------------------------

    // Set decimation mode.  DECIMATION points are thrown out for every one point
    // stored in the A/D FIFO.
    SERVO16_set_adc_decimation(module, DECIMATION);

    // Set additional delay (ns) between A/D and DAC conversion.  This delay
    // should be at least 10800ns + the maximum calculation time.
    SERVO16_set_dac_delay(module, DAC_DELAY);

    // Toggle reset
    SERVO16_reset(module);

    // Enable required channel pairs
    SERVO16_enable_dac_pair(module, PAIR_MASK_OUT);
    SERVO16_enable_adc_pair(module, PAIR_MASK_IN);

    // Set FIFO level threshold so interrupt fires when there
    // is only one sample per channel in the FIFO
    SERVO16_set_adc_fifo_threshold(module, ActivePairsIn-1);

    //-----------------------
    //  Initialize Interrupts
    //-----------------------

    ClearInterrupt(xrpt);
    ActivateInterrupt(xrpt);
    InterruptSource(xrpt, xrpt_source);
    InstallIntVector(analog_isr, xrpt);
    ClearInterrupt(xrpt);
    EnableInterrupt(xrpt);

    printf("\n\n Servo 16 initialized");
}

//-----------------------------------------------------------------------------
//  active_pairs() --  Calculate number of active pairs
//-----------------------------------------------------------------------------

int active_pairs_in()
{
    volatile int pairs = 0;
    volatile unsigned int mask = PAIR_MASK_IN;

    while(mask)
        {
        if(mask & 1)
            pairs++;
        mask >>= 1;
        }

    return pairs;
}

int active_pairs_out()
{
    volatile int pairs = 0;
    volatile unsigned int mask = PAIR_MASK_OUT;
```

```
    while(mask)
        {
        if(mask & 1)
            pairs++;
        mask >>= 1;
        }

    return pairs;
}

//---------------------------------------------------------------------------
//  analog_isr() --  Analog I/O interrupt routine
//---------------------------------------------------------------------------

void analog_isr()
{

    // Get pointer to base address of Servo16 module
    volatile SERVO16 *servo16 = (SERVO16 *)pModule(module);

    // dma copy samples from A/D FIFO to input buffer - Blocking
    SERVO16_bleed_adc_fifo(module, (uint32*)Input, ActivePairsIn);

    // Servo calculations
    DoServoCalc();

    // dma copy samples from output buffer to DAC FIFO - non-Blocking
    dma_mem_to_port(0, (int*)Output, (int*)&servo16->fifo.both, ActivePairsOut, 0);

    // ack interrupt on the module - *required*
    SERVO16_ack_adc_interrupt(module);
}

//---------------------------------------------------------------------------
//  DoServoCalc() --  Servo calculations
//---------------------------------------------------------------------------

void DoServoCalc()
{
    int k;

    // Update time
    ClockCount++;
    t=ClockCount*ts;

    //-----------------------------------------------------------------------
    // Read input channels (PWM-board introduces a -sign on inputs)
    //-----------------------------------------------------------------------
    S2 = -IN_SCALE*(double)(Input[0].half.low); // Sensor 2 [V] (vertical)

    //-----------------------------------------------------------------------
    // Compute reference trajectory and 2nd derivative
    //-----------------------------------------------------------------------
    Reference();

    rv      = r;
    rvddot  = rddot;


    //-----------------------------------------
    // VERTICAL - Compute errors
    //-----------------------------------------

    x       = atan(S2*50e-6/ys);            // Rotation around x-axis
    eold    = err;

    g1      = gn1 - yy*atan(x);     // Upper gap
    g2      = gn2 + yy*atan(x);     // Lower gap

    err     = x-rv;
    edot    = (err-eold)*freq;
    eint   += 0.5*(err+eold)/freq;

    //-----------------------------------------
    // VERTICAL - Sliding mode control
    //-----------------------------------------

    Ss  =  edot + (labda1+labda2)*err + labda1*labda2*eint;
    Us  =  gk1*(2.0/pi)*atan(Ss/phi1);
    Tx  = (1.0*rvddot - (labda1+labda2)*edot - labda1*labda2*err - Us)/H;
```

```
    Tx += m*g*l*1.0; // Compensate gravity

    if (Tx>=0)
        {
        I1      = (g1+1.0*70.0e-6)*sqrt( Tx/(0.335*7.0e-6));
        I2      = 0;
        }
    else
        {
        I1      = 0;
        I2      = g2*sqrt(-Tx*7.0e+5);
        }

    //-----------------------------------------------
    // NO NEGATIVE CURRENT COMMANDS SHOULD BE PASSED
    //-----------------------------------------------

//  if(I1<0){I1 = 0;};
//  if(I2<0){I2 = 0;};

    //-------------------------------------------------
    // Update output channels if no key has been pressed
    //-------------------------------------------------

if (stopcondition==0)
    {
    Output[0].half.low  =(int)(GAIN_SCALE*(I1)*OUT_SCALE);  // Output 0, pin 48
    Output[0].half.high =(int)(GAIN_SCALE*(I2)*OUT_SCALE);  // Output 1, pin 46
    Output[1].half.low  =(int)(GAIN_SCALE*(0)*OUT_SCALE);   // Output 2, pin 27
    Output[1].half.high =(int)(GAIN_SCALE*(0)*OUT_SCALE);   // Output 3, pin 29
    Output[2].half.low  =(int)((10*Ss)*OUT_SCALE);          // Output 4, pin 23
    Output[2].half.high =(int)((rv*5500.0)*OUT_SCALE);      // Output 5, pin 21

/*
    // Output pulse freq should match controller sample freq

    if(odd)
        {
        Output[2].half.low  =(int)(1*OUT_SCALE);
        odd=0;
        }
    else
        {
        Output[2].half.low  =(int)(0*OUT_SCALE);
        odd=1;
        }
*/
    }

    //-------------------------------------------------------------------------
    // Write 0 to all output channels if key has been pressed
    //-------------------------------------------------------------------------
else
    {
    for (k = 0; k < ActivePairsOut; k++)
        {
        Output[k].both = 0;
        }
    }
}

//-----------------------------------------------------------------------------
//  Reference() -- 'Smooth' step
//-----------------------------------------------------------------------------

void Reference()
{
near static double T0       = 0.001;        // Start time of step reference [s]
near static double Tstep    = 1.0;          // Step duration [s]
near static double Amp      = 1.0;          // Step amplitude [corresponding to sensor voltage]

near double     dt          =  Tstep/4.0;
near double     T1          =  T0+dt;
near double     T2          =  T0+3.0*dt;
near double     Tend        =  T0+Tstep;
near double     jerk        =  Amp/(2*dt*dt*dt);
near double     a02         =  jerk*dt;
near double     a03         = -a02;
near double     v02         =  jerk*dt*dt/2.0;
```

```
near double     v03         =  v02;
near double     x02         =  jerk*dt*dt*dt/6.0;
near double     x03         =  jerk*dt*dt*dt*(11.0/6.0);

    if(t<T0)
            {
            rddot   = 0;
            V_INIT = -9.0;
            r       = V_INIT;
            }
    else if( (t>=T0) && (t< T1 ) )
            {
            rddot   =  jerk*(t-T0);
            r       =  jerk*(t-T0)*(t-T0)*(t-T0)/6 + V_INIT;
            }
    else if( (t>=T1) && (t<T2 ) )
            {
            rddot   = -jerk*(t-T1) + a02;
            r       = -jerk*(t-T1)*(t-T1)*(t-T1)/6 + a02*(t-T1)*(t-T1)/2 + v02*(t-T1) + x02 + V_INIT;
            }
    else if( (t>=T2) && (t<Tend ) )
            {
            rddot   =  jerk*(t-T2) + a03;
            r       =  jerk*(t-T2)*(t-T2)*(t-T2)/6 + a03*(t-T2)*(t-T2)/2 + v03*(t-T2) + x03 + V_INIT;
            }
    else if( (t>=Tend) && (t<T02) )
            {
            rddot   = 0;
            r       = V_INIT+Amp;
            }

    // Step defined in [V] for more practical use,
    // now compute corresponding angle [degrees]

    rddot   = (rddot/5500.0);//*50e-6)/ys;  // [ V/s^2 to degrees/s^2 ]
    r       = atan((r*50e-6)/ys);       // [ V to degrees ]
}
```

C.3 6-DOF model (S-function)

```
//===== longstrokeMIMO ============================================================
//      E.Visser, June 2004
//      6 DOF model of tray. Includes motion of linear motors.
//=================================================================================

#define S_FUNCTION_NAME longstrokeMIMO
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#include "math.h"


#define     CONT_ST         24              // No. of continous states:
                                            // 0 - 3   Translation linear motors 1-4 in [m]
                                            // 4 - 9   Tray DOF's x, y, z in [m], theta_x, theta_y, theta_z in [deg]
                                            // 10-11   Short range translations xS and yS [m]
                                            // 12-23   Derivatives of state 0-11
                                            //
                                            // STATES ARE DEPENDENT:
                                            // x0 = x2 (= xL)
                                            // x1 = x3 (= yL)
                                            // x4 = x0 + x10 (= xL + xS)
                                            // x5 = x1 + x11 (= yL + yS)

#define     DISC_ST          0              // No. of discrete states
#define     GLOBALINTS       0              // No. of global integers
#define     GLOBALREALS     16              // No. of global reals
                                            // 0       Gap 11 [m]
                                            // 1       Gap 12 [m]
                                            // 2       Gap 13 [m]
                                            // 3       Gap 14 [m]
                                            // 4       Gap 21 [m]
                                            // ...     ...
                                            // 14      Gap 43 [m]
                                            // 15      Gap 44 [m]

#define     NIP              2              // No. of input ports
                                            // 0       Torque command linear motors [V]
                                            // 1       Coil currents [A]

#define     IPW0             4              // Input port 0 width
                                            // 0 - 3   Torque command linear motor 1-4

#define     IPW1            16              // Input port 1 width
                                            // 0       Current to coil 11
                                            // 1       Current to coil 12
                                            // 2       Current to coil 13
                                            // 3       Current to coil 14
                                            // 4       Current to coil 21
                                            // ...     ...
                                            // 14      Current to coil 14
                                            // 15      Current to coil 15

#define     NOP              4              // No. of output ports
                                            // 0       Translations linear motors [m]
                                            // 1       6 DOF's of tray [m],[deg]
                                            // 2       Physically sensed gaps [m]
                                            // 3       Short stokes xS and yS [m]

#define     OPW0             4              // Output port 0 width
                                            // 0 - 3   Translations linear motors 1-4

#define     OPW1             6              // Output port 1 width
                                            // 0 - 5   Tray DOF's x, y, z, theta_x, theta_y, theta_z

#define     OPW2             8              // Output port 2 width
                                            // 0       Gap 11
                                            // 1       Gap 21
                                            // 2       Gap 32
                                            // 3       Gap 42
                                            // 4       Gap 14
                                            // 5       Gap 24
                                            // 6       Gap 34
                                            // 7       Gap 44

#define     OPW3             2              // Output port 3 width
                                            // 0       xS
```

```
                                        // 1       yS

#define     U(element)      (*uPtrs0[element])  //  Pointer to Input Port0 (Torque command lin. motors)
#define     I(element)      (*uPtrs1[element])  //  Pointer to Input Port1 (Coil currents)


//=============================================================================
// System parameters
//=============================================================================

// Tray:
#define     Jxx         0.5364                  // [kg*m^2]     Inertia tray
#define     Jyy         0.536                   // [kg*m^2]     Inertia tray
#define     Jzz         1.0708                  // [kg*m^2]     Inertia tray
#define     L0          0.55                    // [m]          Nominal arm length
#define     mt          10.9                    // [kg]         Mass tray
#define     gc          9.81                    // [m/s^2]      Gravity constant

// Linear motors:
#define     klin        18.96                   // [N/V]        Force constant linear motor (1..4)
#define     mlin        3.8                     // [kg]         Mass of linear motor (1..4)

// Coils:
#define     gnx         3.81e-4   //3.5e-4      // [m]          Nominal gap size, horizontal (gny = gnx)
#define     gnz         6.35e-4   //6.0e-4      // [m]          Nominal gap size, vertical
#define     kc          1.00e-5   //1.5e-5      // [N*m^2/A^2]  Force constant of magnetic actuator
#define     z_INIT      0//-6.33e-4               // [m]          Initial vertical position (tray rests on lower coils)
                                                //              Maintain gap of 2e-6 [m] (otherwise dividing by 0)
//int_T initpos = 1;                              //              Boolean variable, tray is in initial position TRUE/FALSE

//=============================================================================
// Initialize
//=============================================================================

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);   // Number of expected parameters
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; // Parameter mismatch will be reported by Simulink
    }

    ssSetNumContStates(S, CONT_ST);
    ssSetNumDiscStates(S, DISC_ST);

    if (!ssSetNumInputPorts(S, NIP)) return;
    ssSetInputPortWidth(S, 0, IPW0);
    ssSetInputPortWidth(S, 1, IPW1);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S, NOP)) return;
    ssSetOutputPortWidth(S, 0, OPW0);
    ssSetOutputPortWidth(S, 1, OPW1);
    ssSetOutputPortWidth(S, 2, OPW2);
    ssSetOutputPortWidth(S, 3, OPW3);

    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, GLOBALREALS);
    ssSetNumIWork(S, GLOBALINTS);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

    // Take care when specifying exception free code - see sfuntmpl_doc.c
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}


//=============================================================================
// Initialize continuous states & global real variables
//=============================================================================

#define MDL_INITIALIZE_CONDITIONS

static void mdlInitializeConditions(SimStruct *S)
```

```
{
    real_T *x0 = ssGetContStates(S);
    real_T *G  = ssGetRWork(S);
    real_T  L01,L02,L03,L04;    // Initial arm lengths [m]
    int_T   i;

    x0[0]  =  0;                // Translation motor 1 (x_1)
    x0[1]  =  0;                // Translation motor 2 (y_1)
    x0[2]  =  0;                // Translation motor 3 (x_2)
    x0[3]  =  0;                // Translation motor 4 (y_2)
    x0[4]  =  0;                // Translation tray    (x)
    x0[5]  =  0;                // Translation tray    (y)
    x0[6]  =  z_INIT;           // Translation tray    (z)
    x0[7]  =  0;                // Rotation tray       (theta_x)
    x0[8]  =  0;                // Rotation tray       (theta_y)
    x0[9]  =  0;                // Rotation tray       (theta_z)
    x0[10] =  0;                // Short stroke xS
    x0[11] =  0;                // Short stroke yS

    for(i=0;i<12;i++)
        {
        x0[12+i] = 0;           // Derivatives of state 0-11
        }

    // Compute initial arm lenghts (ignore short strokes)
    L01 = L0+x0[5];
    L02 = L0+x0[4];
    L03 = L0-x0[5];
    L04 = L0-x0[4];

    // Compute initial gap sizes:
    G[0]  = gnx   - ((x0[4]-x0[0]) + L01*tan(x0[9]));
    G[1]  = 2*gnx - G[0];
    G[2]  = gnz   - ( x0[6]        - L01*tan(x0[7]));
    G[3]  = 2*gnz - G[2];

    G[4]  = gnx   - ((x0[5]-x0[1]) - L02*tan(x0[9]));
    G[5]  = 2*gnx - G[4];
    G[6]  = gnz   - ( x0[6]        - L02*tan(x0[8]));
    G[7]  = 2*gnz - G[6];

    G[8]  = gnx   - ((x0[4]-x0[2]) - L03*tan(x0[9]));
    G[9]  = 2*gnx - G[8];
    G[10] = gnz   - ( x0[6]        + L03*tan(x0[7]));
    G[11] = 2*gnz - G[10];

    G[12] = gnx   - ((x0[5]-x0[3]) + L04*tan(x0[9]));
    G[13] = 2*gnx - G[12];
    G[14] = gnz   - ( x0[6]        + L04*tan(x0[8]));
    G[15] = 2*gnz - G[14];
}


//=============================================================================
// Define outputs
//=============================================================================

static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T             *TLM = ssGetOutputPortRealSignal(S,0);
    real_T             *DOF = ssGetOutputPortRealSignal(S,1);
    real_T             *GAP = ssGetOutputPortRealSignal(S,2);
    real_T             *SHS = ssGetOutputPortRealSignal(S,3);
    real_T             *x   = ssGetContStates(S);
    real_T             *G   = ssGetRWork(S);
    int_T              i;

    UNUSED_ARG(tid);            // Not used in single tasking mode

    for(i=0;i<4;i++)            // Output port 0 = translations of linear motors 1-4
        {
        TLM[i] = x[i];
        }

    for(i=0;i<6;i++)            // Output port 1 = 6 DOF's of tray
        {
        DOF[i] = x[i+4];
        }
                                // Output port 2 = 8 gaps that are physically sensed
    GAP[0] = G[0];              // Gap 11
```

```
    GAP[1] = G[4];              // Gap 21
    GAP[2] = G[9];              // Gap 32
    GAP[3] = G[13];             // Gap 42
    GAP[4] = G[3];              // Gap 14
    GAP[5] = G[7];              // Gap 24
    GAP[6] = G[11];             // Gap 34
    GAP[7] = G[15];             // Gap 44

    SHS[0]= x[10];              // xS
    SHS[1]= x[11];              // yS

  }


//=============================================================================
// Compute derivatives of continuous states
//=============================================================================

#define MDL_DERIVATIVES

static void mdlDerivatives(SimStruct *S)
{
    InputRealPtrsType uPtrs0 = ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType uPtrs1 = ssGetInputPortRealSignalPtrs(S,1);
    real_T  *dx              = ssGetdX(S);
    real_T  *x               = ssGetContStates(S);
    real_T  *G               = ssGetRWork(S);
    real_T  F[16];              // Magnetic forces
    real_T  FL[4];              // Forces driving linear motors
    real_T  L1,L2,L3,L4;        // Arm length [m]
    int_T   i;                  // Index
    real_T  KC[16];             // Force constant per coil
    real_T  sumFZ;              // Sum of forces in z-direction

    static real_T  wkc[16] = {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
    for (i=0;i<16;i++)
            {
            KC[i] = kc*wkc[i];  // Multiplicative perturbation of coil force constants
            }
    /*
    // Compute arm lenghts
    L1 = L0+x[5]; // L0+y
    L2 = L0+x[4]; // L0+x
    L3 = L0-x[5]; // L0-y
    L4 = L0-x[4]; // L0-x

    // Compute gaps and store in matrix
    G[0] = gnx   - ((x[4]-x[0])  + L1*tan(x[9]));
    G[1] = 2*gnx - G[0];
    G[2] = gnz   - ( x[6]        - L1*tan(x[7]));
    G[3] = 2*gnz - G[2];

    G[4] = gnx   - ((x[5]-x[1])  - L2*tan(x[9]));
    G[5] = 2*gnx - G[4];
    G[6] = gnz   - ( x[6]        + L2*tan(x[8]));
    G[7] = 2*gnz - G[6];

    G[8] = gnx   - ((x[4]-x[2])  - L3*tan(x[9]));
    G[9] = 2*gnx - G[8];
    G[10] = gnz   - ( x[6]       + L3*tan(x[7]));
    G[11] = 2*gnz -G[10];

    G[12] = gnx   - ((x[5]-x[3]) + L4*tan(x[9]));
    G[13] = 2*gnx - G[12];
    G[14] = gnz   - ( x[6]       - L4*tan(x[8]));
    G[15] = 2*gnz - G[14];

    // Show error message if any gap is (smaller than) 0
    for (i=0;i<16;i++)
            {
            if (G[i]<1e-6)
                {
                ssSetErrorStatus(S,"Gap 0: MECHANICAL CONTACT!");
                }
            }

    // Compute magnetic forces
    for (i=0;i<16;i++)
            {
            F[i] = KC[i]*I(i)*I(i)/(G[i]*G[i]);
```

```
        }

    // Compute forces driving linear motors
    for (i=0;i<4;i++)
        {
        FL[i] = klin*U(i);
        }
    */


    // Compute 1st derivatives
    for (i=0;i<12;i++)
        {
        dx[i] = x[i+12];
        }

    // Compute 2nd derivatives

    // Translations of linear motors:
    dx[12] = (U(0)-U(2))/mlin;      //(FL[0]+F[1] -F[0] )/mlin;
    dx[13] = 0;//U(1)/mlin;      //(FL[1]+F[5] -F[4] )/mlin;
    dx[14] = dx[10];        // Slave to motor 1    // (FL[2]+F[9] -F[8] )/mlin;
    dx[15] = dx[11];        // Slave to motor 2    // (FL[3]+F[13]-F[12])/mlin;

    // Translations of tray:
    dx[16] = U(2)/mt;       //(F[0]-F[1]+F[8]-F[9])/mt;
    dx[17] = 0;//U(3)/mt;       //(F[4]-F[5]+F[12]-F[13])/mt;

    /*
    sumFZ = (F[2]-F[3]+F[6]-F[7]+F[10]-F[11]+F[14]-F[15]-mt*gc);
    if((sumFZ<=0)&(x[6]<=-6.3e-4))//&(initpos==1))
        {
        dx[18]=0;
        }
    else
        {
        dx[18] = sumFZ/mt;
        //initpos=0;
        }
     */

     dx[18]=0;

    // Rotations of tray:
    dx[19] = 0; //(-L1*(F[2]-F[3]) +L3*(F[10]-F[11]))/Jxx;
    dx[20] = 0; //( L2*(F[6]-F[7]) -L4*(F[14]-F[15]))/Jyy;
    dx[21] = 0; //( L1*(F[0]-F[1]) -L2*(F[4]-F[5]) -L3*(F[8]-F[9]) + L4*(F[12]-F[13]))/Jzz;

    // Short strokes:
   // dx[22] = dx[14]-dx[10];     // xS_ddot
    dx[22]  = U(0)/mlin - ((mlin+mt)/(mlin*mt))*U(2);

    dx[23] = dx[15]-dx[11];     // yS_ddot

}


//=============================================================================
// Terminate
//=============================================================================

static void mdlTerminate(SimStruct *S)
{
    UNUSED_ARG(S); /* unused input argument */
}

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

C.4 MIMO-controller (S-function)

```
//===== SMC6DOF.c ================================================================
//      E.Visser, June 2004
//      Sliding mode control of tray
//================================================================================

#define     S_FUNCTION_NAME  longstrokeSMC
#define     S_FUNCTION_LEVEL 2

#include    "simstruc.h"
#include    "math.h"

#define     U(element) (*uPtrs[element])        //              Pointer to Input Port0
#define     CONT_ST     0                       //              No. of continuous states
#define     DISC_ST     0                       //              No. of discrete states
#define     GLOBALINTS  0                       //              No. of global integers
#define     GLOBALREALS 0                       //              No. of global reals


//================================================================================
// System parameters
//================================================================================

// Tray:
#define     Jxx         0.5364                  // [kg*m^2]     Inertia tray
#define     Jyy         0.536                   // [kg*m^2]     Inertia tray
#define     Jzz         1.0708                  // [kg*m^2]     Inertia tray
#define     L0          0.55                    // [m]          Nominal arm length
#define     mt          10.9                    // [kg]         Mass tray
#define     gc          9.81                    // [m/s^2]      Gravity constant

// Coils:
#define     gnx         3.81e-4                 // [m]          Nominal gap size, horizontal (gny = gnx)
#define     gnz         6.35e-4                 // [m]          Nominal gap size, vertical
#define     kc          1.00e-5                 // [N*m^2/A^2]  Force constant of magnetic actuator

// Other:
#define     Ts          2.5e-4                  // [s]          Sample time
#define     pi          3.14159265358979


//================================================================================
// Initialize
//================================================================================

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
        {
        return;     // Parameter mismatch will be reported by Simulink
        }

    ssSetNumContStates(S,CONT_ST);                  // set number of continuous states
    ssSetNumDiscStates(S,DISC_ST);                  // set number of discrete states

    if (!ssSetNumInputPorts(S, 6)) return;          // 4 inputs (x,r,rddot and gaps)
    ssSetInputPortWidth(S, 0, 6);                   // Measured states
    ssSetInputPortWidth(S, 1, 6);                   // Reference
    ssSetInputPortWidth(S, 2, 6);                   // Second derivative of reference
    ssSetInputPortWidth(S, 3, 8);                   // Gaps g11,g21,g32,g42,g14,g24,g34,g44
    ssSetInputPortWidth(S, 4, 6);                   // Derivative of errors
    ssSetInputPortWidth(S, 5, 6);                   // Integral of errors
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S, 2)) return;
    ssSetOutputPortWidth(S, 0, 4);                  // Output to linear motors
    ssSetOutputPortWidth(S, 1, 16);                 // Current to coils

    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S,GLOBALREALS);                   // set number of global reals
    ssSetNumIWork(S,GLOBALINTS);                    // set number of global ints
    ssSetNumPWork(S,0);                             // set number of pointer work vector elements
    ssSetNumModes(S,0);                             // set number of operating modes for the block
    ssSetNumNonsampledZCs(S,0);                     // set number of nonsampled zero crossings that block detects
 }

static void mdlInitializeSampleTimes(SimStruct *S)
```

```
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}


//=============================================================================
// Control computation and output
//=============================================================================

static void mdlOutputs(SimStruct *S, int_T tid)
{
    InputRealPtrsType uPtrs     = ssGetInputPortRealSignalPtrs(S,0);
    real_T      *y              = ssGetOutputPortRealSignal(S,0);

// Parameters sliding mode algorithm:
    real_T  LABDA[6] = {25,    25,    10,     15,     15,     20};      // Labda (for x, y, z, theta_x, theta_y, theta_z)
    real_T    PHI[6] = {1e-2,  1e-2,   1e-2,   7e-3,   7e-3,   1e-2};   // Phi
    real_T     GK[6] = { 10,    10,     15,     10,     10,     15};     // Gain of switching term

    real_T KI2= (4.3656e-4)/1.24e-9;
    real_T KP2= (3.1549e-5)/1.24e-9;
    real_T KD2= (1.1921e-7)/1.24e-9;
    real_T klin = 18.96;
    real_T mlin = 3.8;
    real_T kscale= mt*klin/mlin;

// For mapping of Fcog to Fact (see 'Compute_M.m')
    static real_T Ms[8][6]  = { { 0.5, 0  , 0   , 0      , 0      , 0.4545 },
                                { 0  , 0.5, 0   , 0      , 0      , -0.4545 },
                                { 0.5, 0  , 0   , 0      , 0      , -0.4545 },
                                { 0  , 0.5, 0   , 0      , 0      , 0.4545 },
                                { 0  , 0  , 0.25, -0.9091 , 0      , 0      },
                                { 0  , 0  , 0.25, 0      , 0.9091 , 0      },
                                { 0  , 0  , 0.25, 0.9091 , 0      , 0      },
                                { 0  , 0  , 0.25, 0      , -0.9091 , 0      }
                              };

// Non-zero entries ( = diagonal elements) of B-matrix:
    static real_T H[6] = {0.0917, 0.0917, 0.0917, 1.8643, 1.8643 , 0.9339};  // ={1/mt, 1/mt, 1/mt, 1/Jxx, 1/Jyy, 1/Jzz};


    real_T Ss[6];       // Sliding surface for 6 DOF's
    real_T Us[6];       // Switching terms
    real_T Fc[6];       // Control action (force/torque) w.r.t. COG
    real_T FHact[4];    // Control action mapped to horizontal actuators
    real_T FVact[4];    // Control action mapped to vertical actuators
    real_T gap  [4][4]; // Gaps
    int_T  i;


// Compute control action for 6 DOF's
    for (i=0;i<6;i++)
            {
            // Compute Sliding surface
            Ss[i]     =  U(i+26) + 2.0*LABDA[i]*(U(i)-U(i+6)) + LABDA[i]*LABDA[i]*U(i+32);
            // Compute switching term
            Us[i]     =  GK[i]*(2.0/pi)*atan(Ss[i]/PHI[i]);
            // Compute control signal (force/torque) w.r.t. COG
            Fc[i]     = (U(i+12) - (LABDA[i]*LABDA[i]*(U(i)-U(i+6)) + 2.0*LABDA[i]*U(i+26) + Us[i])) / H[i];
            }


// Compensate force in z-direction for gravity
    Fc[2]=Fc[2]+mt*gc;


// Map control action to actuators
    for(i=0;i<4;i++)    // Horizontal
            {
            FHact[i]    = Ms[i][0]*Fc[0]+Ms[i][1]*Fc[1]+Ms[i][2]*Fc[2]+Ms[i][3]*Fc[3]+Ms[i][4]*Fc[4]+Ms[i][5]*Fc[5];
            }
    for(i=4;i<8;i++)    // Vertical
            {
            FVact[i-4]  = Ms[i][0]*Fc[0]+Ms[i][1]*Fc[1]+Ms[i][2]*Fc[2]+Ms[i][3]*Fc[3]+Ms[i][4]*Fc[4]+Ms[i][5]*Fc[5];
            }


    // Gap measurements from capacitive sensors:
    gap[0][0] = U(18);    // g11
```

```
    gap[1][0] = U(19);    // g21
    gap[2][1] = U(20);    // g32
    gap[3][1] = U(21);    // g42
    gap[0][3] = U(22);    // g14
    gap[1][3] = U(23);    // g24
    gap[2][3] = U(24);    // g34
    gap[3][3] = U(25);    // g44

    // Compute complementary gaps:
    gap[0][1] = 2*gnx - gap[0][0];
    gap[0][2] = 2*gnz - gap[0][3];
    gap[1][1] = 2*gnx - gap[1][0];
    gap[1][2] = 2*gnz - gap[i][3];
    gap[2][0] = 2*gnx - gap[2][1];
    gap[2][2] = 2*gnz - gap[2][3];
    gap[3][0] = 2*gnx - gap[3][1];
    gap[3][2] = 2*gnz - gap[3][3];

    // Input to linear motors: DONT SENT TO LINEAR MOTORS BUT USE FOR FEED FORWARD!!
    y[0] = 0;
    y[1] = 0;
  //  y[2] = Fc[0]; // Fx

    y[2] = kscale * (KD2 * U(26) + KP2* (-U(0)+U(6)) + KI2 * U(32));
    y[3] = Fc[1]; // Fy

    // Translate actuator forces to input currents (using inverted non-linearity, non-robust)
    for(i=0;i<4;i++)    // Horizontal coils
            {
            if (FHact[i]>0)
                {
                y[4+(4*i)] = gap[i][0]*sqrt( FHact[i]/kc);
                y[5+(4*i)] = 0;
                }
            else
                {
                y[4+(4*i)] = 0;
                y[5+(4*i)] = gap[i][1]*sqrt(-FHact[i]/kc);
                }
            }

    for(i=0;i<4;i++)    // Vertical coils
            {
            if (FVact[i]>0)
                {
                y[6+(4*i)] = gap[i][2]*sqrt( FVact[i]/kc);
                y[7+(4*i)] = 0;
                }
            else
                {
                y[6+(4*i)] = 0;
                y[7+(4*i)] = gap[i][3]*sqrt(-FVact[i]/kc);
                }
            }
}


//=============================================================================
// Terminate
//=============================================================================

static void mdlTerminate(SimStruct *S)
{
}

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

C.5 Demo program

The following code has been used to demonstrate some basic functionalities of the linear motors and Unidex software. It contains initialization, basic motions like translations and circular motion of the middle point of the tray, independent operation or master/slave operation of the linear motors, and some examples of WHILE- and FOR-loops in G-code.

For more details and advanced options, refer to [3].

```
PROGRAM METRIC INCREMENTS        ;Use metric system

WAIT AL                          ;Wait until all previous commands are executed
ENABLE X Y Z U                   ;Enable all axes

MESSAGE DISPLAY "Axis X, Y, Z and U enabled"

HOME X Y Z U                     ;Home all axes

GEAR 3,1,1,1                     ;Link axis Z as slave to axis X at 1:1 ratio
GEAR 4,2,1,1                     ;Link axis U as slave to axis Y at 1:1 ratio

;i=0
;WHILE (i<10)                    ;While loop example (not used)
; Enter commands here
;i=i+1
;ENDWHILE

G1 x10 y10 f600                  ;Move 10 mm away from home position, feedrate 600 mm/s

:REPEAT

LOOP 5                           ;Loop 5 times, forward and back over 180 mm
G1 x180 y180 f30000
G1 x-180 y-180 f30000
NEXT

GEAR 3,0                         ;Decouple Z from X
GEAR 4,0                         ;Decouple U from Y

G1 y180 z180 f30000              ;Move Y and Z to opposite end of range

DWELL 1000                       ;Dwell 1000 milliseconds

G1 x180 f25000                   ;Motors 'chase' each other
G1 u180 f25000
G1 z-180 f25000
G1 y-180 f25000
G1 y180 f25000
G1 z180 f25000
G1 u-180 f25000
G1 x-180 f25000

DWELL 1000

LOOP 1                           ;Motors travel full range in 15 small steps (forward and back)
LOOP 15
G1 x10 y-10 z-10 u10 f20000
DWELL 300
NEXT
LOOP 15
G1 x-10 y10 z10 u-10 f20000
DWELL 300
NEXT
NEXT

DWELL 1000

G1 y-180 z-180 f20000            ;Bring Y and Z back to same positions as X and U, resp.

GEAR 3,1,1,1                     ;Link slaves to masters again
GEAR 4,2,1,1

G1 x100 y10 f20000
DWELL 1000

CW X0 Y0 I0 J75                  ;Clockwise cirkel, start point (0,0) mm and middle point (0,75)
                                 ;w.r.t. current position of middle point of all axes
```

```
DWELL 1000

CCW X0 Y0 I0 J75                ;Counter-clockwise cirkel
DWELL 1000

G1 x-100 y-10 f20000

DWELL 2000

GOTO :REPEAT                    ;Go to REPEAT, infinitely loop this demo
                                ;Commend out to use following code

;AT X,1,50,10                   ;Autotuning command generates sinusoidal reference
;AT Y,1,50,10                   ;with increasing frequency and decreasing amplitude
;AT Z,1,50,10
;AT U,1,50,10

;DISABLE X Y                    ;Disable masters

;MESSAGE DISPLAY "DWELL 5 SEC"
;DWELL 5000                     ;Manually move masters, and slaves will track

;GEAR 3,0
;GEAR 4,0
;DISABLE U Z

;DI X Y Z U
;MESSAGE DISPLAY "Axis X, Y, Z and U disabled"

EXIT
```

Appendix D

M-files

D.1 Fit model to measurement data

```
close all;clear all;clc;
% load all measured forces
load meas2308conditioned;

% Geometry
a   = 1.753*25.4e-3;            % [m]
b   = 0.455*25.4e-3;            % [m]
c   = 0.631*25.4e-3;            % [m]
d   = 0.433*25.4e-3;            % [m]
e   = 2.628*25.4e-3;            % [m]
f   = 0.648*25.4e-3;            % [m]
g   = 0.325*25.4e-3;            % [m]
h   = 0.5*25.4e-3;              % [m]
i   = (0.5*e-0.5*d)*25.4e-3;    % [m]
j   = (a-0.5*b)*25.4e-3;        % [m]
k   = 0.272*25.4e-3;            % [m]

S1  = c*f;  %[m^2]
S2  = d*f;  %[m^2]
S3  = b*f;  %[m^2]
S4  = h*f;  %[m^2]

% Constants
N   = 258;
mu0 = 4*pi*1e-7;
mur = 47000;    % From Thesis L.Fevre p.17
mu  = mu0*mur;

R1  = k/(mu*S1) + i/(mu*S3) + j/(mu*S2);
R2  = i/(mu*S4);
R3  = g/(mu*S1);

A   = (S1+2*S2)/(2*S1*S2);

% Sensor values for different air gaps
x = [ -8:5];

% Compute air gap
for i=1:length(I);
    for m=1:14
    Itot(m,i)=I(i);
    Xtot(m,i)= (9.50-x(m))*50e-6*0.335/0.275;
    end
end

% Force measurements [V]
Fsens = [F8; F7; F6; F5; F4; F3; F2; F1; F0; Fplus1; Fplus2; Fplus3; Fplus4; Fplus5];
% Force [N]
Fmeas = Fsens/10.1;

% Equilibrium of moments:
% 0.335*Fact + Fmeas*0.405 = 1.69*9.81*0.245 + 8.0382*9.81*0.4;

% used load arm is 0.37. This is a calculated value to get the balance of
% moments correct at zero current
```

```
Fact    = ((1.69*9.81*0.245 + 8.0382*9.81*0.375) - Fmeas*0.405) /0.335;

% If arm sticks, force measurement is not valid so measurement value is 0
% [V]. This leads to high (>80 N) actuator force. Remove from data:
for i=1:length(I);
    for m=1:14
        if  Fact(m,i) > 80;
            Fact(m,i) = 0;
        end
    end
end

% Plot measurements
figure
mesh(Xtot,Itot,Fact)
hold on
title('Force measurements')
xlabel('Air gap [m]')
ylabel('Current [A]')
zlabel('Force [N]')

%-------------------------------------------------------------------------
% Fit theoretical model to measurements by adjusting gain and additional
% gap length

% Compute absolute error for different additional gap lenghts and gains:
for gi = 1:100;
    gain(gi)= 0.5+0.01*gi;
    for add_x   = 0:1:300;
        x0 = add_x * 1e-6;
        for r=1:14;
            for ix = 1:length(Xtot);
                Fmodel(r,ix)     = gain(gi) * -0.5*N^2*mu0/A*  1./(mu0/A*(0.5*R1 + 0.5*R2  + R3)...
                                   + x0 + Xtot(r,ix)).^2 * Itot(r,ix).^2;
                if  Fact(r,ix) == 0;
                    Fmodel(r,ix) = 0;
                end
                err(r,ix)       = -Fact(r,ix)-Fmodel(r,ix);
                err_crit2(r,ix) = sqrt(err(r,ix)^2);
            end
        end
        G0(gi,add_x+1)  = gain(gi);
        X0(gi,add_x+1)  = x0;
        E2(gi,add_x+1)  = sum(sum(err_crit2));
    end
end

% Plot error as function of gain and x0
figure
mesh(X0,G0,E2)
grid on
xlabel('Additional air gap [m]')
ylabel('Gain [-]')
zlabel('Force [N]')
title('Sum of absolute error as function of x_0 and gain')

% Idem, zoom. For better visualization don't plot values larger than 225
for gi = 1:100;
    for add_x   = 0:1:300;
        if E2(gi,add_x+1) >= 225;
           E2(gi,add_x+1) = 225;
       end
    end
end

figure
mesh(X0,G0,E2)
grid on
axis([0 300e-6 0.5 1.5 205 225])
xlabel('Additional air gap [m]')
ylabel('Gain [-]')
zlabel('Force [N]')
title('Sum of absolute error as function of x_0 and gain - ZOOMED')

% Find minimum value. Minimum value for every column:
[Ecolmin,Row_index]=min(E2);
% Minimum value of row Y
[Emin,Col_index]=min(Ecolmin)
% Check
E2(Row_index(Col_index), Col_index)
```

```
% Return optimum values:
gain    = G0(Row_index(Col_index), Col_index)
x0      = X0(Row_index(Col_index), Col_index)

%-------------------------------------------------------------------------

% Compute model with optimum parameter values and plot
clear Fmodel

% Calculate model with optimum values for gain and x0
for ix=1:14;
    for ii=1:length(Itot);
        Fmodel(ix,ii) = gain * 0.5*N^2*mu0/A*  1./(mu0/A*(0.5*R1 + 0.5*R2  + R3)...
                        + x0 + Xtot(ix,ii)).^2 * Itot(ix,ii).^2;
    end
end

% where actuator force was not measurable and made 0, make model force 0 as well
for i=1:length(I);
    for m=1:14
        if  Fact(m,i) == 0;
            Fmodel(m,i) = 0;
        end
    end
end

% Plot measurement and model
figure
mesh(Xtot,Itot,Fact)
hold on
mesh(Xtot,Itot,Fmodel)
title('Force measurement and model')
xlabel('Air gap [m]')
ylabel('Current [A]')
zlabel('Force [N]')

% Plot difference between model and measurements
figure
mesh(Xtot,Itot,Fact-Fmodel)
hold on
title('Force measurement minus model')
xlabel('Air gap [m]')
ylabel('Current [A]')
zlabel('Force [N]')
```

D.2 Closed loop system ID

```
% ------------------------------------------
% System ID for arm with 1 DOF (vertical)
% Derive linear model from closed loop system ID
% ------------------------------------------
close all;clear all;clc
load CLIDsim2.mat;       % Contains u, v, w (and test_e, test_u)
fs = 4000;
Npoints=2^12;           % 4096;
Noverlap=round(0.9*Npoints);

[SensCoh,F] = cohere(w,u,Npoints,fs,hanning(Npoints),Noverlap);
[SensFRF,F] =    tfe(w,u,Npoints,fs,hanning(Npoints),Noverlap);

[CLCoh,F] = cohere(r,thx,Npoints,fs,hanning(Npoints),Noverlap);
[CLFRF,F] =    tfe(r,thx,Npoints,fs,hanning(Npoints),Noverlap);

figure
subplot(2,1,1)
semilogx(F,20*log10(abs(SensFRF)),'k')
grid
ylabel('Mag [dB]','fontsize',16)
title('Sensitivity','fontsize',16)
subplot(2,1,2)
semilogx(F,angle(SensFRF)/pi*180,'k')
grid
ylabel('Phase [deg]','fontsize',16)
xlabel('Frequency [Hz]','fontsize',16)

figure
semilogx(F,SensCoh,'k')
grid
xlabel('Coherence [-]','fontsize',16)
xlabel('Frequency [Hz]','fontsize',16)
title('Coherence of sensitivity measurement','fontsize',16)

figure
subplot(2,1,1)
semilogx(F,20*log10(abs(CLFRF)),'k')
grid
ylabel('Mag [dB]','fontsize',16)
title('Closed loop - Simulated','fontsize',16)
subplot(2,1,2)
semilogx(F,angle(CLFRF)/pi*180,'k')
grid
ylabel('Phase [deg]','fontsize',16)
xlabel('Frequency [Hz]','fontsize',16)

figure
semilogx(F,CLCoh,'k')
grid
xlabel('Coherence [-]','fontsize',16)
xlabel('Frequency [Hz]','fontsize',16)
title('Coherence of closed loop measurement','fontsize',16)

% Controller used in closed loop
a = [ 0.09478           0;    9.605e-005           1];
b = [ 9.605e-005;  1.633e-008];
c = [  -6.828e+009    1.48e+009];
d = [ 7.5e+005];
C_PIDdisc=ss(a,b,c,d,2.5e-4);
C_PID1 = freqresp(C_PIDdisc,2*pi*F);
for i=1:2049;
    C_PID(i,1)=C_PID1(:,:,i);
end
```

```
% Derive open loop and plant from sensitivity
OpenLoop = 1./SensFRF-1;
Plant    = OpenLoop./C_PID;

% Linearization of system model
Jxx =   0.3897;                                 %// [kg*m^2]
ma  =   1.7327;                                 %// [kg]
g   =   9.81;                                   %// [m/kg*s^2]
l   =   0.2456;                                 %// [m]
yy  =   0.3353;                                 %// [m]
K1  =   4.1603e-6;                              %// [N*m^2/A^2]
K2  =   4.1603e-6;                              %// [N*m^2/A^2]
gnz =   6.3500e-4;                              %// [m]
In1 =   1.0;                                    %// [A]
In2 =   1.0;%gnz * sqrt ( (K1/K2)*In1^2/gnz^2);  %// [A]
Ts  =   1/fs;                                   %// [s]

A  = [  0                                                       1  ;
        (2*yy^2*K1*In1^2/gnz^3+2*yy^2*K2*In2^2/gnz^3)/Jxx       0 ];

B  = [  0                                                  ;
        (2*yy*K1*In1/gnz^2+2*yy*K2*In2/gnz^2)/Jxx ];

C  = [  1   0 ];
D  = [  0 ];

LinSys = ss(A,B,C,D);
figure
%[MagLinSys,PhLinSys] =
bode(LinSys,2*pi*F);

% Plot plant and compare to linearization of model
figure
subplot(2,1,1)
semilogx(F,20*log10(abs(Plant)),'k')
hold on
grid
%semilogx(F,20*log10(abs(MagLinSys)),'r')
ylabel('Mag [dB]','fontsize',16)
title('Plant','fontsize',16)
subplot(2,1,2)
semilogx(F,angle(Plant)/pi*180,'k')
hold on
grid
ylabel('Phase [deg]','fontsize',16)
xlabel('Frequency [Hz]','fontsize',16)
```